Xray Essentials
for Cloud

Training Goals

- Understand Xray features and capabilities
- Understand Xray concepts and entities
- Learn how to use Xray to design, organize, plan and execute tests

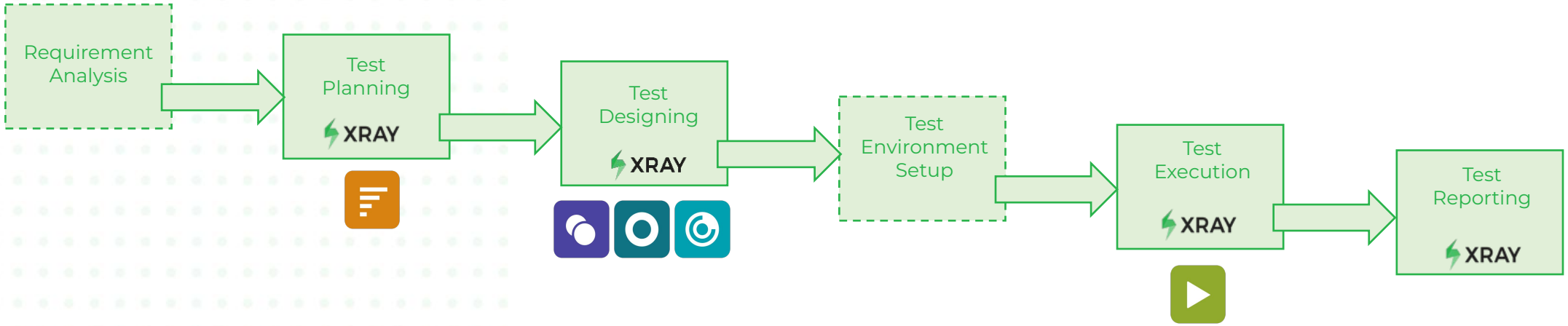
Content

Test Management Essentials

Software Testing Life Cycle

- Plan phase: Use Test Plan issues.
- Design phase: To specify use Precondition and Test issues.

  To organize using Test Sets or the Test Repository.
- Execute phase: Use Test Execution issues.
- Report phase: Use Test Execution issues, built-in requirement coverage reports and custom ones using Jira gadgets.



Requirement Coverage

A requirement is considered:

- COVERED: when at least one Test is associated to it
- UNCOVERED: if no Test is associated to it

We can also say that, on a given scope such as a version, a covered requirement is:

- OK: if all linked Tests are passing
- FAIL or NOK: if at least one of the linked Tests are failing
- TODO: if the linked Tests are yet pending execution

From another point of view, a covered requirement is:

- COMPLETE: if all linked Tests are passing
- INCOMPLETE: if at least one of the linked Tests is failing

Xray Features

Xray Features

- Uses Jira issues for Test Management
- Allows workflows and screens customization
- Manages manual and automated Tests, including Cucumber
- Manages Data Sets to allow Data-Driven Testing
- Promotes test case reusability and composition with Modular Tests

- Allows execution of manual Tests and stores execution results in Jira
- Ensures consistency between Test Executions and changes on Test Specifications
- Tracks changes in all test artefacts

Xray Features

- Tracks tests and requirements in real-time
- Provides reporting in Jira dashboards
- Provides beautiful customizable reports with Document Generator

- Provides complete API (REST API + GraphQL)
- Integrates with CI tools
- Imports data from other systems

Xray Features

- Allows custom test and step statuses
- Provides extensive configuration options
- Major languages supported (English, German, French, Spanish)

- Widely used by many organizations, including Fortune 500 companies
- Always up-to-date with frequent releases and new features
- Ready to use: install and use it right away

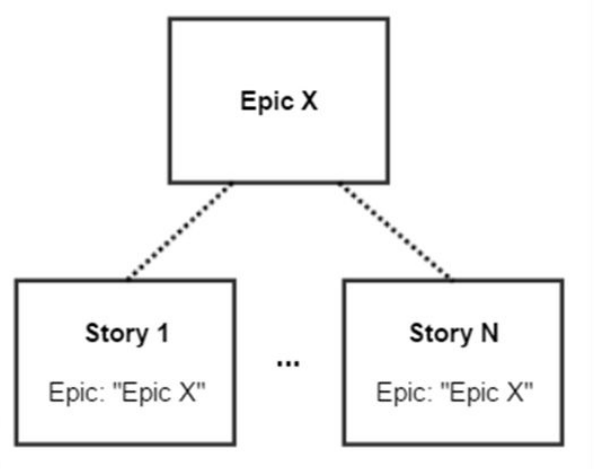
Requirement Coverage Status in Xray

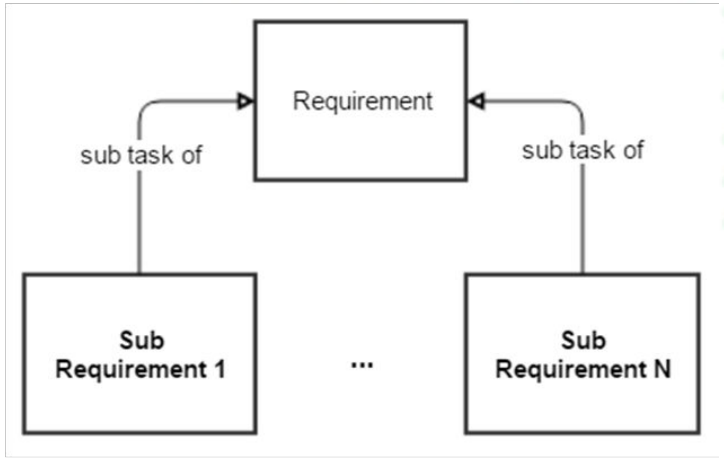
There are 5 possible statuses for coverage

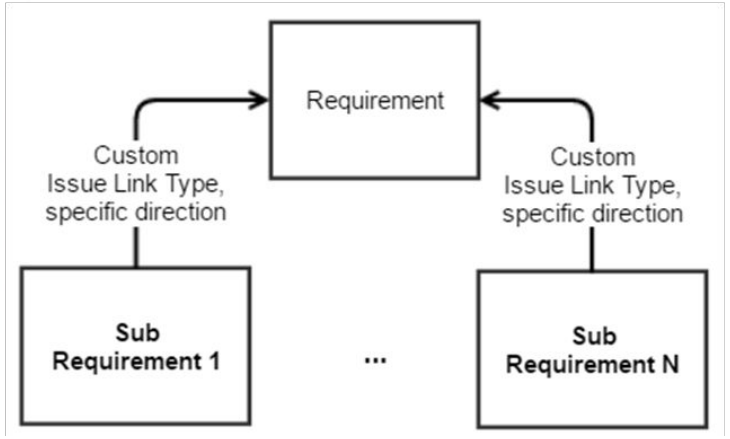
The calculated value depends on the tests and their results for some scope of analysis.

OK	All the tests associated with the issue (e.g. requirement) are **PASSED**
NOK	At least one test associated with the issue is **FAILED**
NOTRUN	At least one test associated with the issue is **TODO**, **ABORTED** or EXECUTING and there are no tests with status **FAILED**
UNKNOWN	At least one test associated with the issue is **UNKNOWN** and there are no Tests with status **FAILED**
UNCOVERED	The issue has no tests associated to it

Sub-Requirements in Xray







Sub-Requirements in Xray

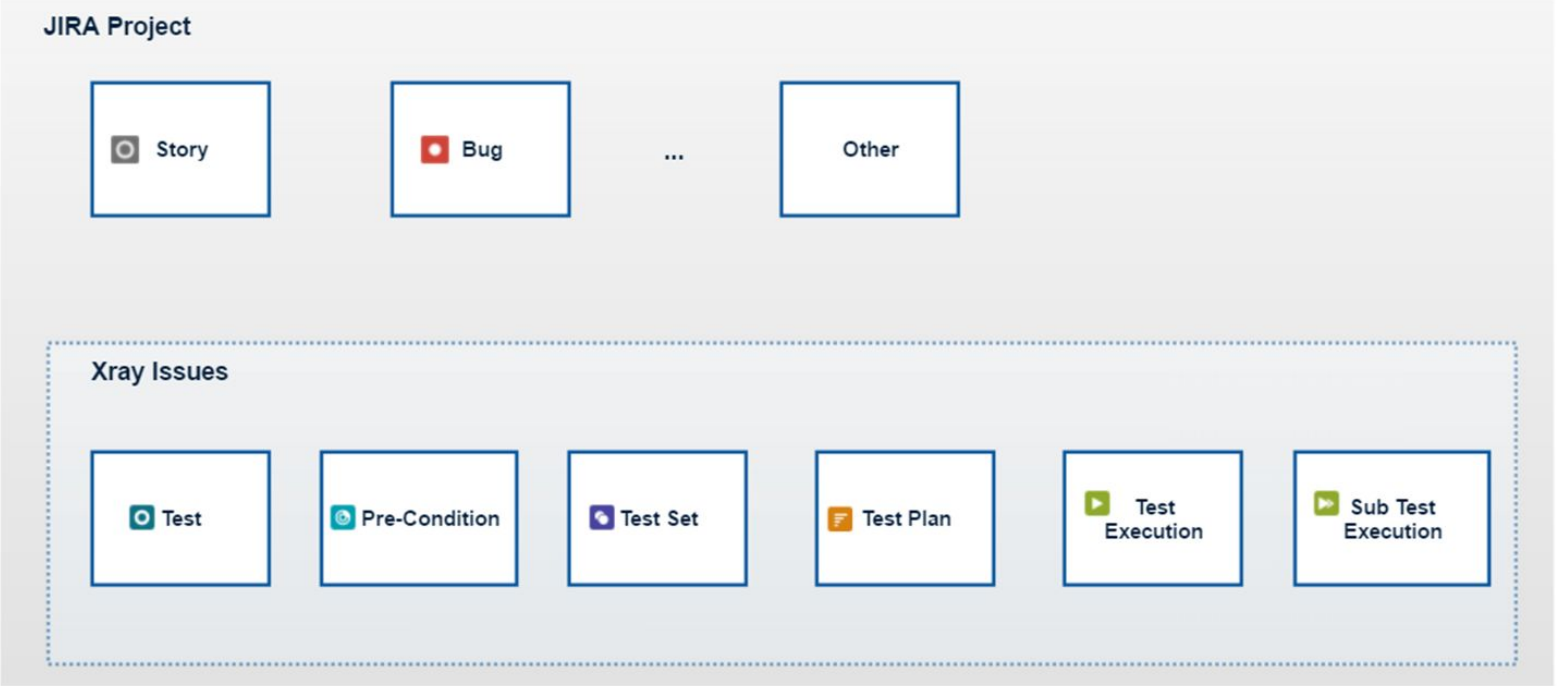
The Requirement status will take into account the Sub-Requirements status.

REQ ∧ SUB-REQ	OK	NOK	NOT RUN	UNKNOWN	UNCOVERED
OK	OK	NOK	NOT RUN	UNKNOWN	OK
NOK	NOK	NOK	NOK	NOK	NOK
NOT RUN	NOT RUN	NOK	NOT RUN	UNKNOWN	NOT RUN
UNKNOWN	UNKNOWN	NOK	UNKNOWN	UNKNOWN	UNKNOWN
UNCOVERED	OK	NOK	NOT RUN	UNKNOWN	UNCOVERED

Project Organization

All for One and One for All

A single project to manage your Requirements and Defects, Test related issues and also have all your Test Executions.



All for One and One for All

Pros

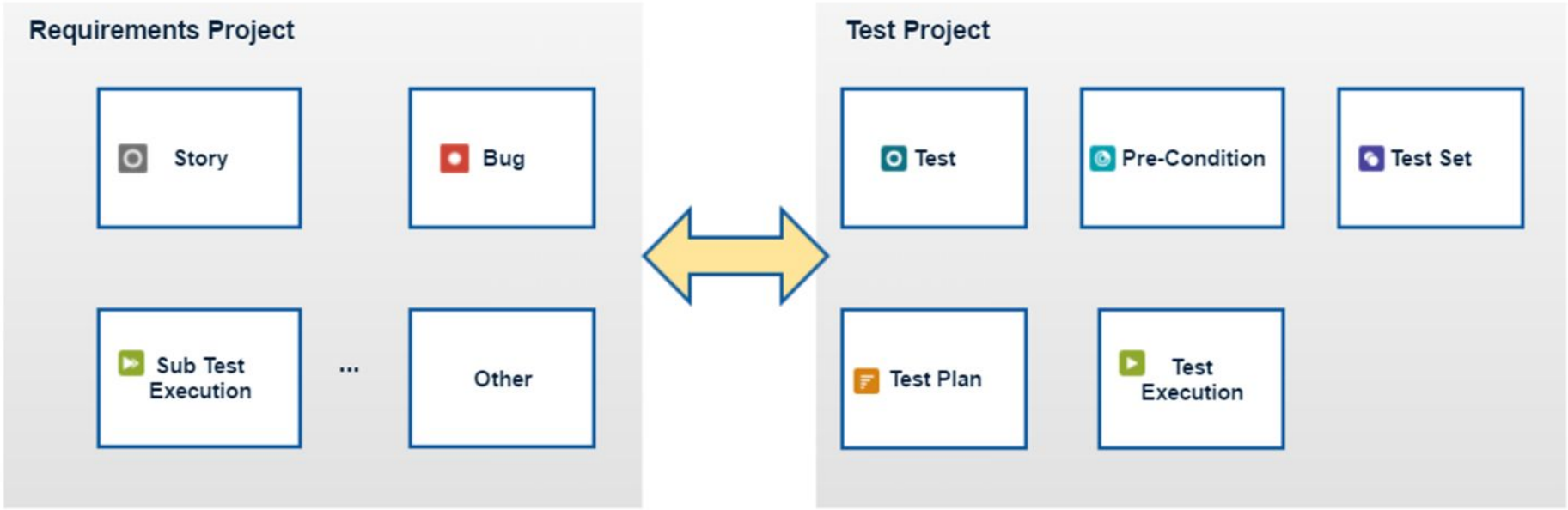
- Self-contained
- Easy to understand and manage
- Promotes team collaboration

Cons

- You may not want to have Bugs, Test Executions in the same project for security concerns

Separate my Requirements and Defects from Tests

If you already have a Jira project for managing requirements and defects and don't want to have any tests nor executions in this project, you can create a separate companion project just for testing purposes.



Don't mix my Requirements and Defects with Tests

Pros

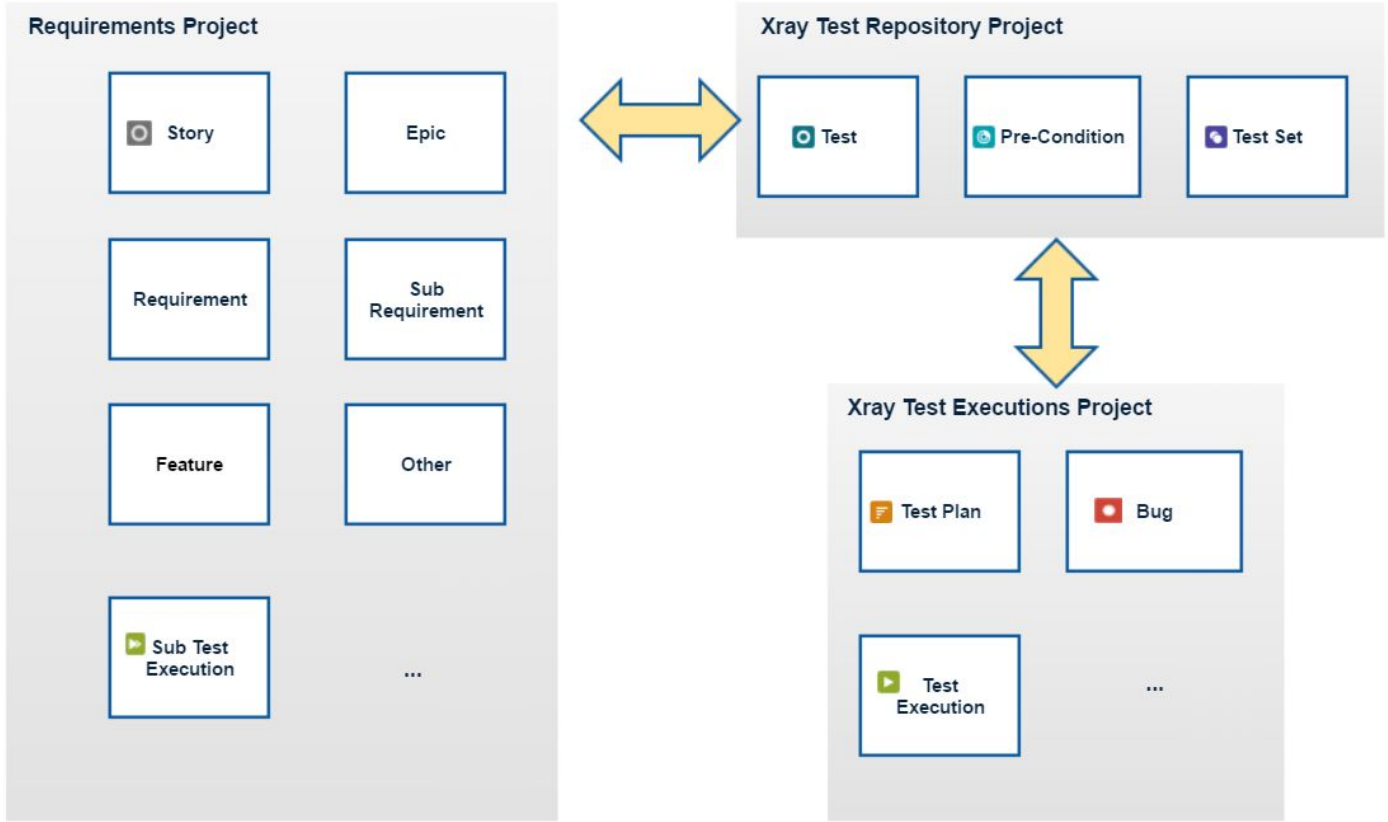
- This separation allows to manage permissions more effectively
- Keeps your existing requirements and defects project as-is, completely separated from testing related tasks

Cons

- Requires manually synchronizing versions from the requirements project to the other ones
- More projects to manage
- May promote team separation

Dedicated Test Repository

A dedicated project for Test and Test Set issues. Test Executions are responsibility of other project(s).



Dedicated Test Repository

Pros

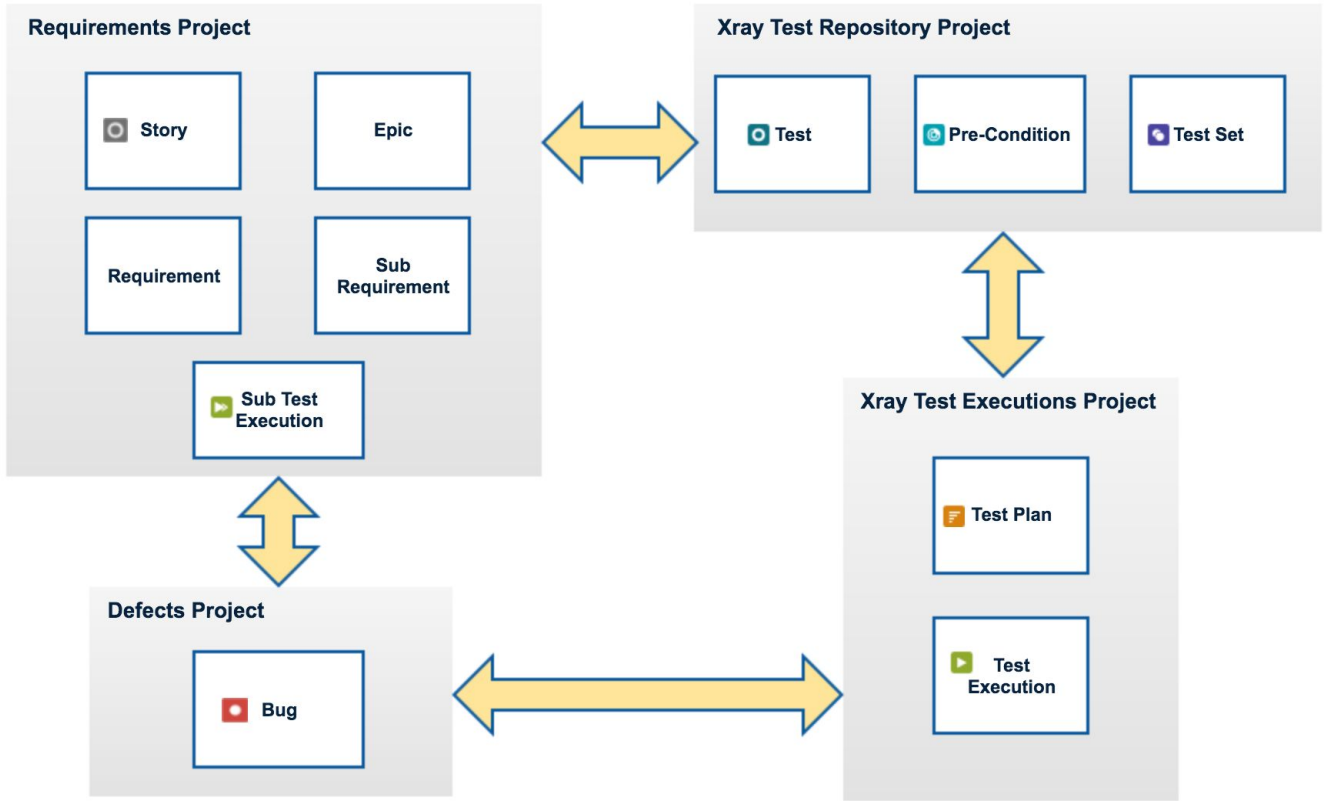
- This separation allows to manage permissions more effectively, so maybe only a set of testers has permission to write tests and others to execute
- Keeps your existing requirements and defects project as-is, completely separated from testing related tasks
- Isolates Test specification and organization
- Keeps sensible data related with bugs and test planning/execution in a different project

Cons

- Requires manually synchronizing versions from the requirements project to the other ones
- More projects to manage
- May promote team separation

Completely Separated

The Test repository, Requirements, Executions and Defects all separated and being handled on different projects.



Completely Separated

Pros

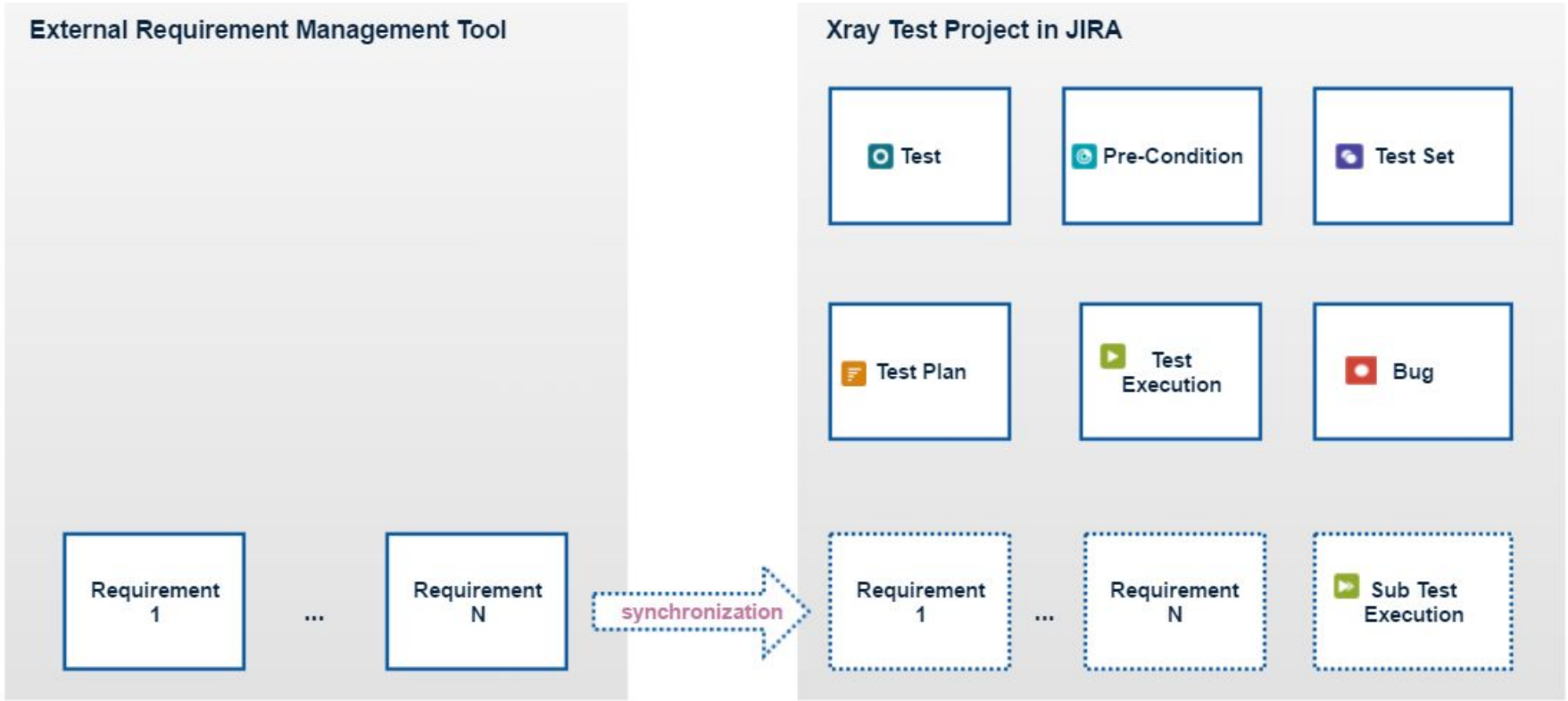
- This separation allows to manage permissions and visibility more effectively
- Keeps your existing requirements and defects project as-is, completely separated from testing related tasks
- Isolates everything

Cons

- Requires manually synchronizing versions from the requirements project to the other ones
- More projects to manage
- May promote team separation

Isolated Repository for Tests

The requirements are outside Jira. A synch process is recommended.



Isolated Repository for Tests

Pros

- Can be used in an initial stage whenever migrating requirements from other tools

Cons

- You will loose the benefits of requirement coverage and traceability

Wrap-up of Project Organization Possibilities

- The ”all-in-one” strategy is simpler to manage and self-contained
- Think on your security/visibility needs, namely if you want to have separate projects for some issue types
- Are your tests going to cover requirements from several different projects? If so, then you may manage those tests in a separate project
- Having Test Executions and Test Plans on different projects will require synchronizing project versions

Xray Concepts and Entities

Test

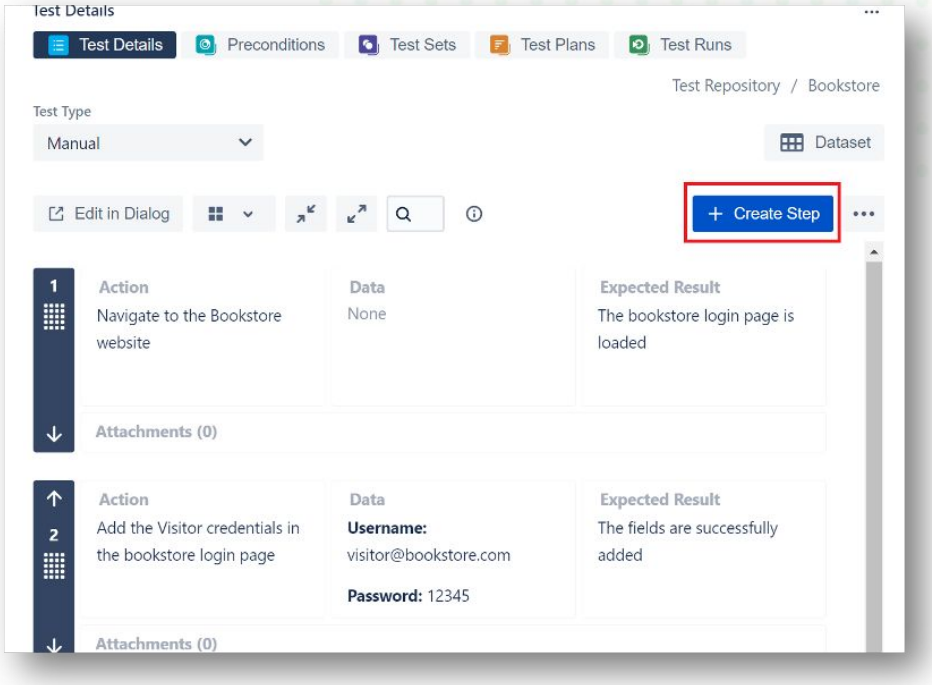
An abstraction of a test idea/scenario, and thus reusable. It is essentially a way to verify/validate the associated requirement(s).

- Can be a scripted (e.g. test case, automated test) or exploratory test
- Can be specified using Gherkin (Scenario)
- Can be executed manually or through automation
- May be linked to/cover one or more requirements
- As any other Jira issue type, it can be labelled, prioritized, assigned to components, commented, etc.

Test – Manual

A traditional test case composed by a list of steps, thus scripted.

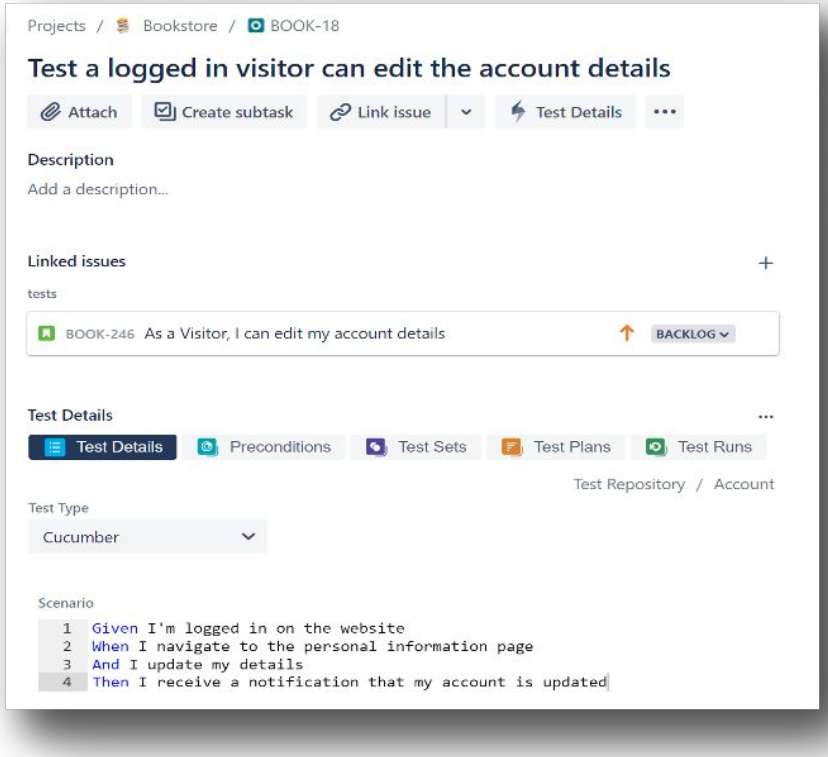
- Each step is defined by:
  - Action/step
  - Data
  - Expected result
  - Input attachments
  - Optionally, test step custom fields
- Support for Jira Wiki markup
- Ability to import steps, (including from Excel) using *copy/paste*



Test – Cucumber

A Cucumber Scenario/Scenario Outline that provides one or more examples, respectively, of an acceptance criterion.

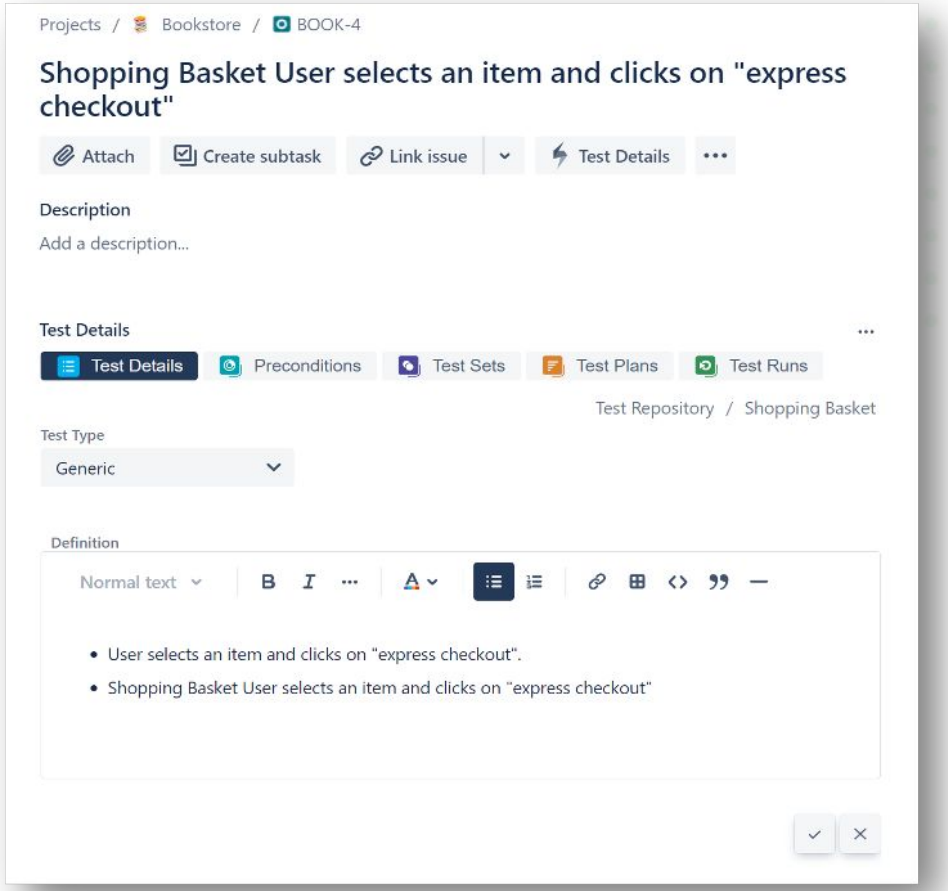
- Write tests in a business-readable domain-specific language (Gherkin)
- Specify Cucumber Scenarios and Scenario Outlines with Gherkin syntax highlighting
- Ability to export to .feature files and execute during Continuous Integration



Test – Generic

An unstructured test, without steps. A way to abstract and have visibility of traditional automated tests or exploratory tests.

- Allows you to manage automated tests that are non-Cucumber (e.g. JUnit, TestNG, Robot)
  - Automate tests in any framework and report their results to the given Test
- Can also be used for exploratory testing, as means to define its charter and goals



Precondition

Abstracts initial steps that must be done or ensured before starting the test.

- Can be described using Gherkin (Background element)
- Can be executed/ensured manually or through automation
- Reusable: can be linked to multiple Tests

Test Set

A flat ordered list of tests organized by some logical way.

- It's a Jira issue: can be labelled, prioritized, assigned to components, commented, etc.
- Can also be used to cover Requirements directly
- A Test can be part of part of multiple Test Sets
- A Test Set can contain Tests from the same project or from other projects

Test Execution

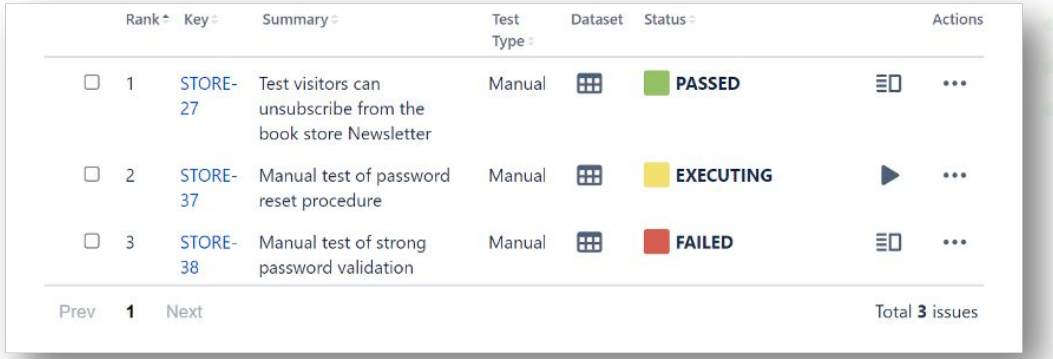
A “task” for executing a group of tests on a given version of the system, on a given environment. This task will also contain the results when they’re reported.

- Contains a list of Tests and their results (i.e. Test Runs)
- Can be labelled, prioritized, assigned to components, etc.
- May be planned or ad-hoc
- May be created manually or during Continuous Integration

Test Run

An instance of a Test in the context of a Test Execution. A run of a Test in some scope.

- Contains the results obtained for a Test, including evidence and linked/reported defects
- For compliance, also contains a copy of the Test specification at the moment of execution
- Assignable (by default inherits assignee from Test Execution issue)
- It's an internal entity; not a Jira issue type



	Rank	Key	Summary	Test Type	Dataset	Status	Actions
☐	1	STORE-27	Test visitors can unsubscribe from the book store Newsletter	Manual		PASSED	
☐	2	STORE-37	Manual test of password reset procedure	Manual		EXECUTING	
☐	3	STORE-38	Manual test of strong password validation	Manual		FAILED	

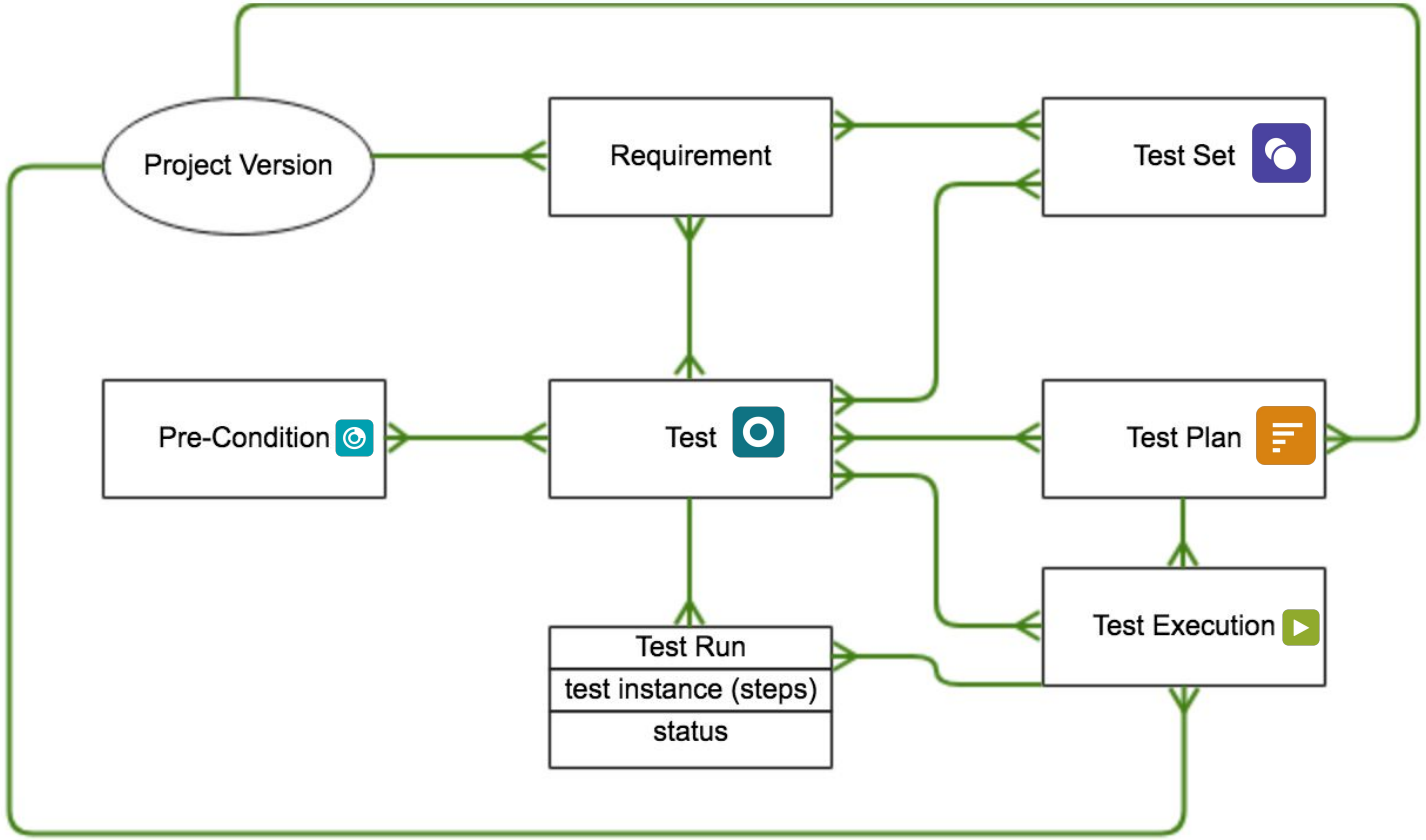
Prev 1 Next Total 3 issues

Test Plan

A way to define the scope for testing and track its progress; what testing we'll be performing and its latest status.

- Tracks a group of tests and their results independently of the number of executions
- Can be used in a planned way, with its scope for testing (i.e. the Tests) defined beforehand
- Can be used in an Agile way, acting as a testing guidance result aggregator, by allowing you to create/add Tests along with their results at any time
- Test Plans may be assigned to versions, sprints and users, as they're a Jira issue
- A version or a sprint may have multiple Test Plans

Xray Entities Relationship Diagram



Although not common, you may link a Test Execution with multiple Test Plans making its results impact the consolidated overview shown/managed on each linked Test Plan.

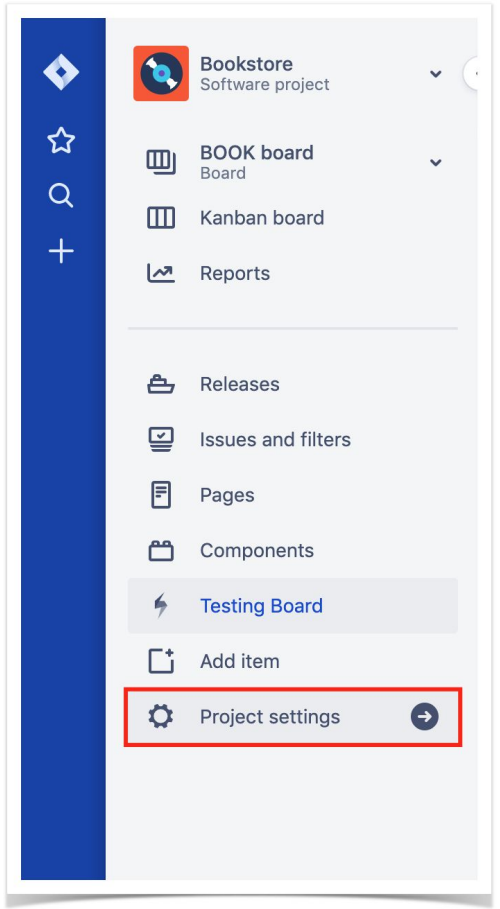
Getting Started

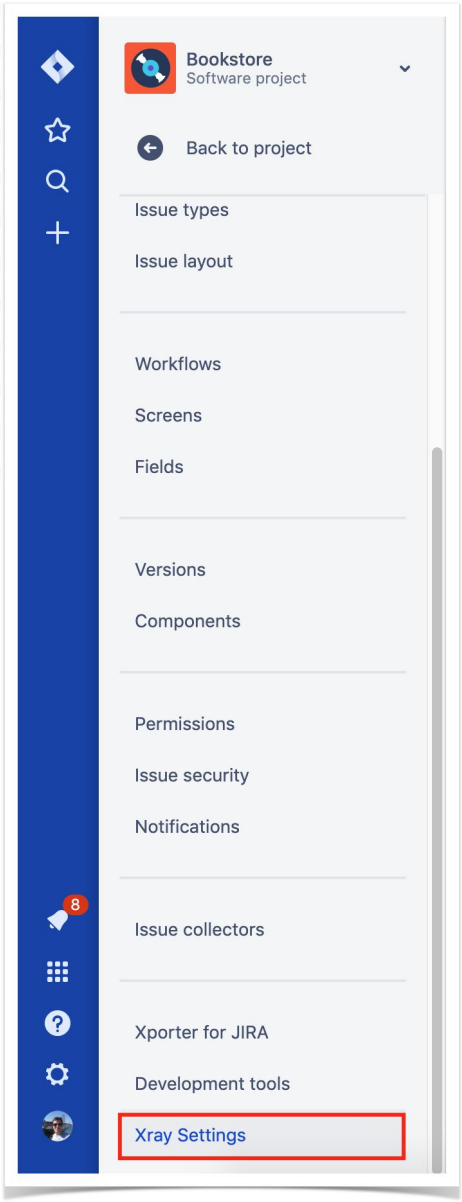
Setting up a project for Xray

After installing Xray, there are basically a few steps you need to Configure Xray Issue Types, Requirements and Defects in your project.

You can do it for a brand-new project or for an existing one.

- These operations can all be done from within **Xray Settings** available at the project settings page.
- You can add Xray issue types (e.g. Test, Test Set, Test Execution, etc.)
- The configuration steps will be different for a next-gen project or in a classic project.



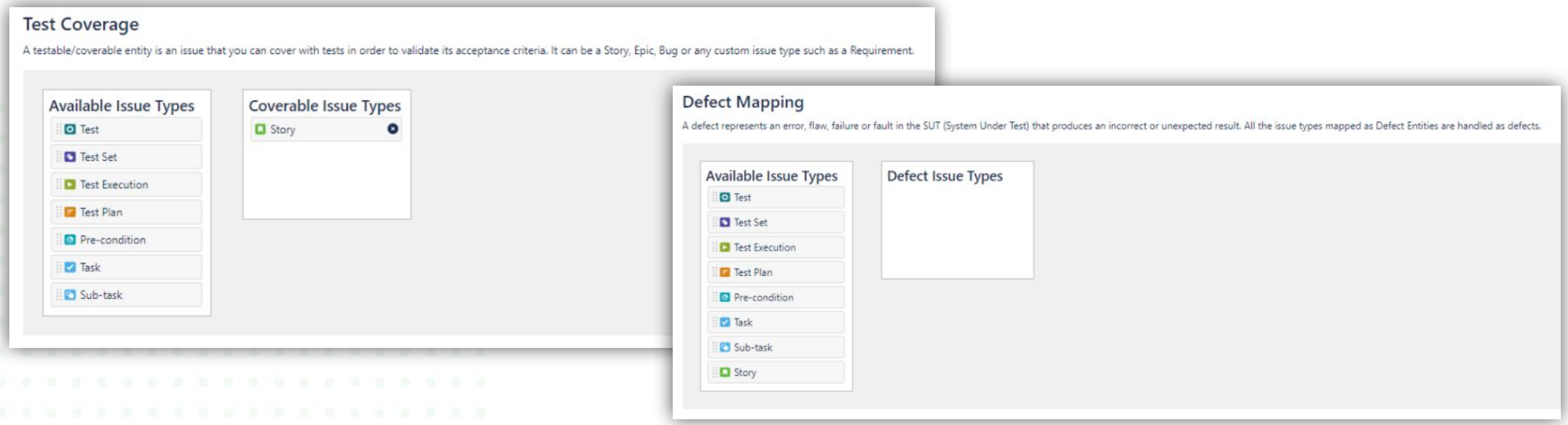


Learn more: https://docs.getxray.app/display/XRAYCLOUD/Quick+Setup

Issue Type Mapping

In Xray's administration, it's possible to define the issue types coverable by Tests (e.g. also known as "requirements") and the ones that can be treated as "defects".

A coverable issue type usually represents a requirement/Story/feature. However, a "Bug" may also be covered by Tests if configured as such.

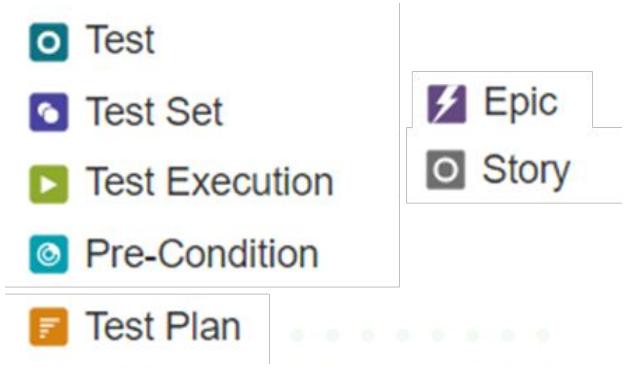


If configured properly, the issue screen on coverable issue types will show coverage information.

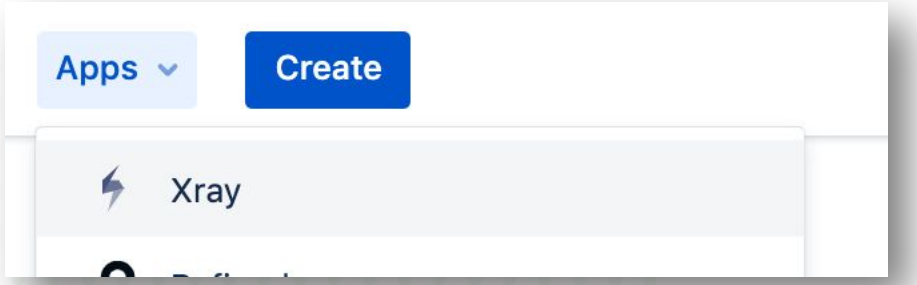
Working with Xray

Presenting the UI interface

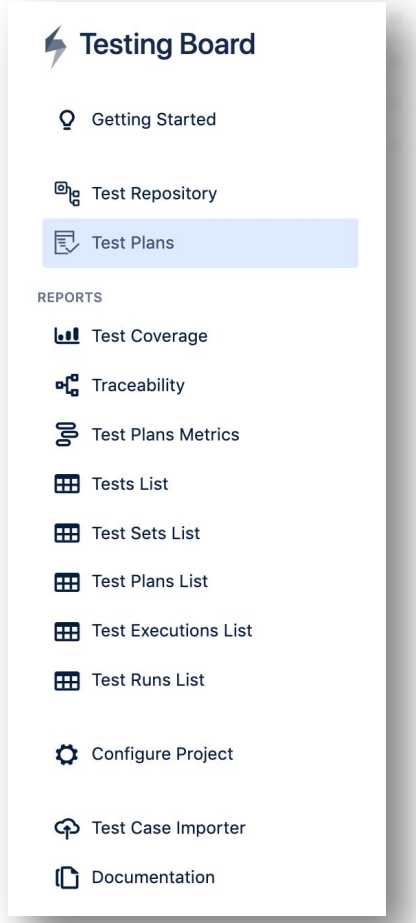
- Specific Jira issues screens



- "Apps > Xray" top menu with shortcuts

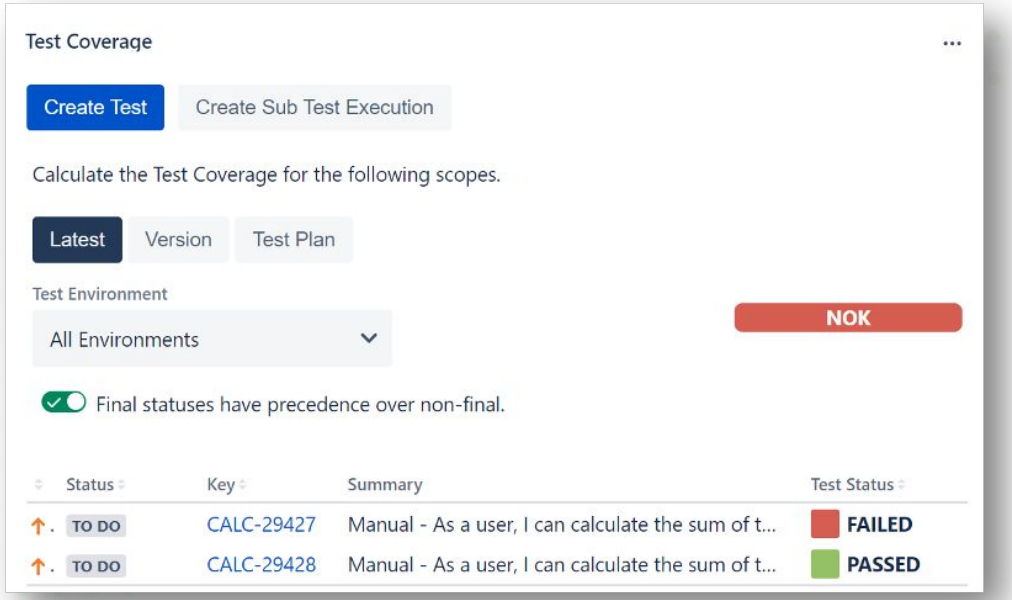


- Testing Board (inside project)



Requirement issue screen

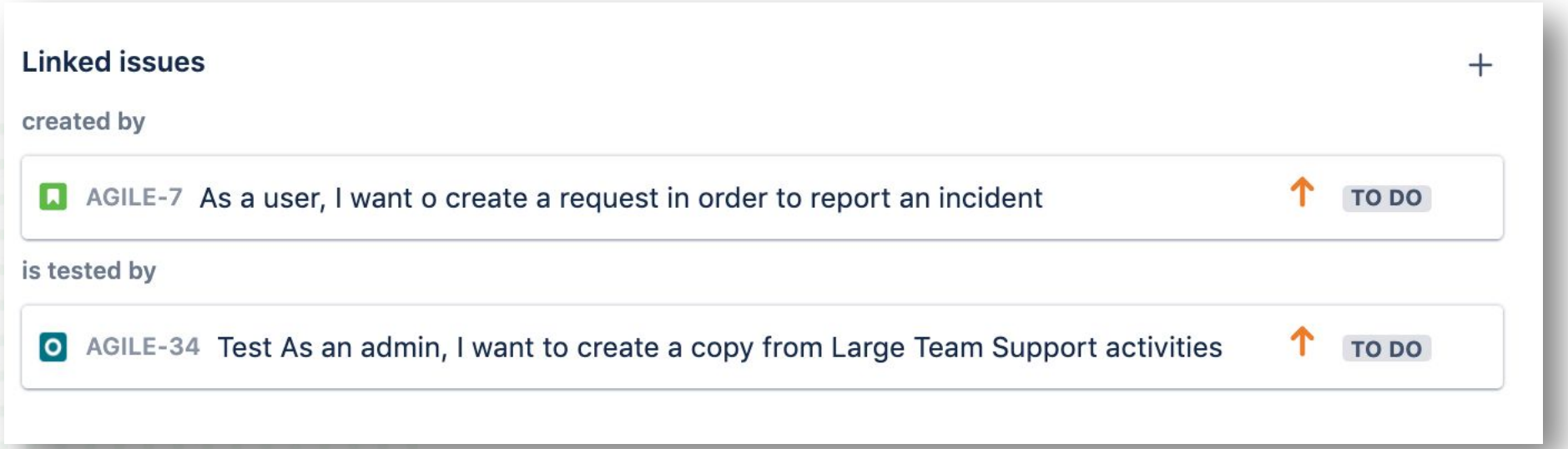
- “Test Coverage” shows coverage status in real-time, according to the behaviour defined in Xray settings and the tests related with that requirement.



- Tests may easily be created and associated to the requirement

Requirement/covered issue screen

- “Linked Issues” show all related issues, including Tests
- Notice that direct coverage is based on the existence of the “is tested by” issue link between the Requirement and the Test issue(s)



Designing Tests

What is a Test?

An abstraction of a test idea/scenario. It is essentially a way to verify or validate the associated requirement(s). It's one or more checks, an experiment, or an investigation towards the goal of obtaining information about some quality attribute of the test target.

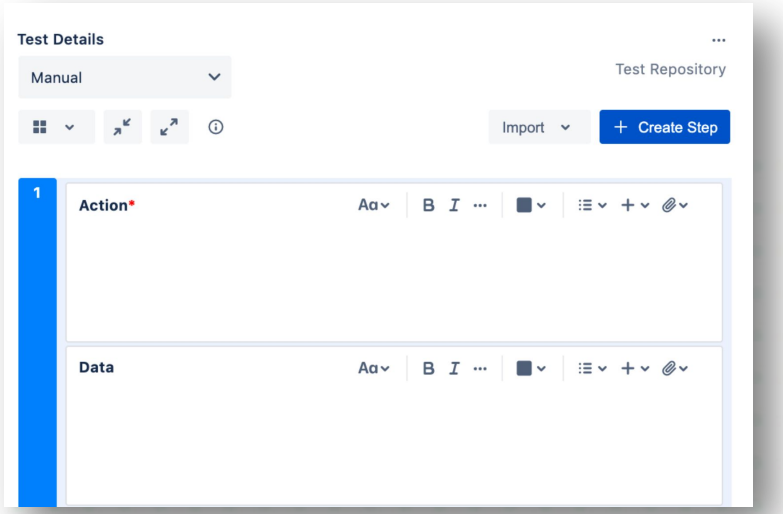
- In a traditional test case executed manually, it's a sequence of steps coupled with conditions or variables, test inputs and an expected results. It is intended to establish the quality, performance or reliability of a piece of system, i.e., test target
- Usually, every requirement or objective the test target is expected to achieve needs at least one Test.
- The success of the Test is determined by comparing expected and actual result

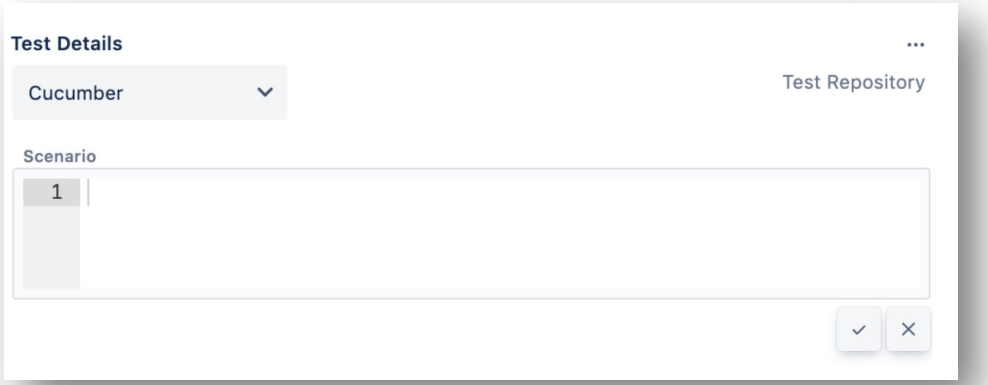
Test Types

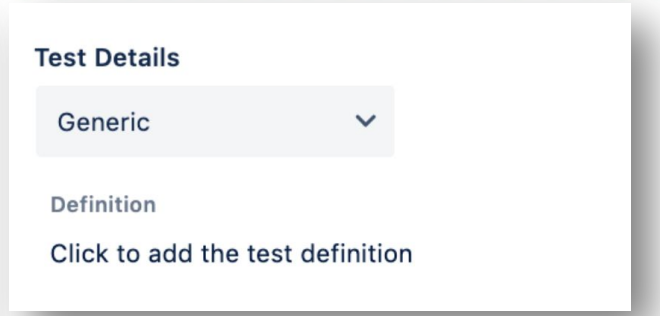
Manual

Cucumber

Generic







By default, Xray provides 3 test types (Manual, Cucumber, Generic), each one of a specific kind (structured, gherkin, unstructured).

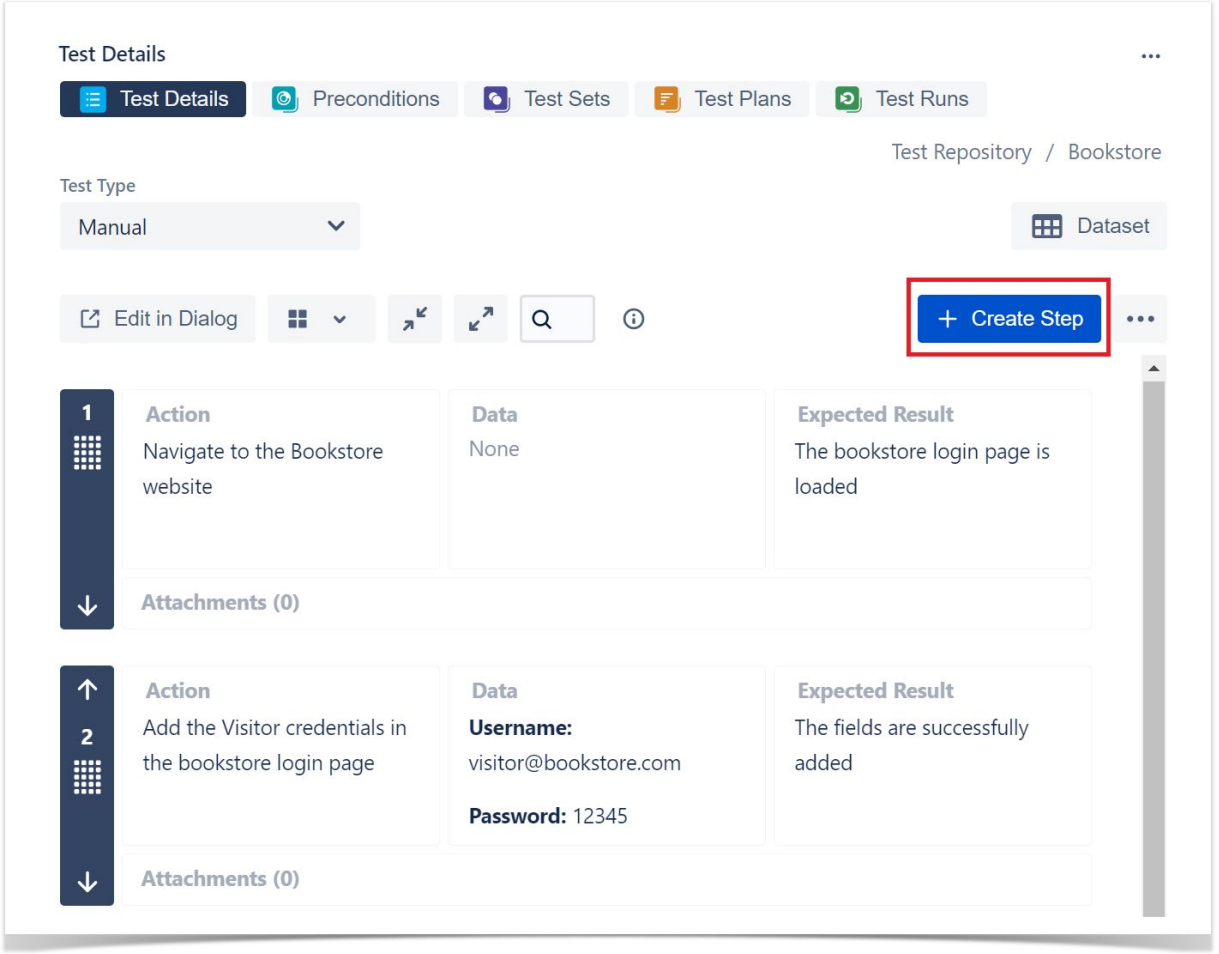
Writing / Specifying Tests

When you create a Test, what needs to be done?

- Specify “Test Type”
- Depending on the test kind, you’ll either add individual steps, define the scenario using Gherkin or fill out the definition with something unique to that test
- Optionally, associate with existing Preconditions
- Link to Requirement(s) (i.e. Story, Epic)

Working with Test Steps

When creating a Manual Test, you can create steps, as if you we're following a script and checking while doing so.

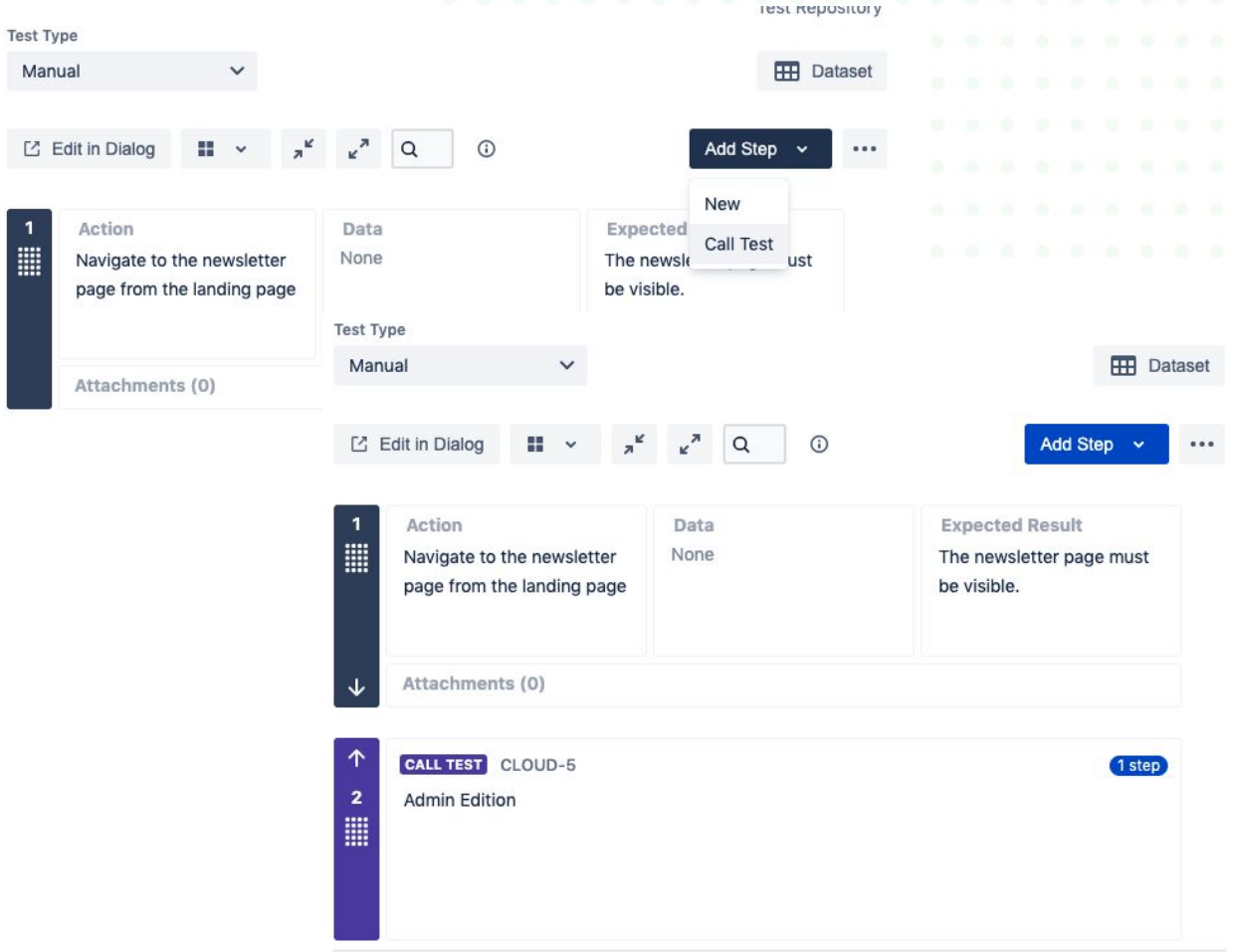


Test Steps allow you to:

- Edit using wiki markup
- Clone, move/rerank, remove steps
- Add attachments to steps
- Import steps from a Test, CSV, JSON or clipboard/Excel

Working with Test Steps - Call Test

When writing tests, you may want to re-use some steps from others tests. Instead of writing the same steps over and over again, you can create a test case composed of other test cases. To achieve this you will need to use the **Call Test feature.**



Call Test allows you to:

- Reuse a previous test specification
- Create composed Tests with a depth limit of 5 called tests

*In the image, Called Tests are represent in purple.

Create Precondition

Precondition

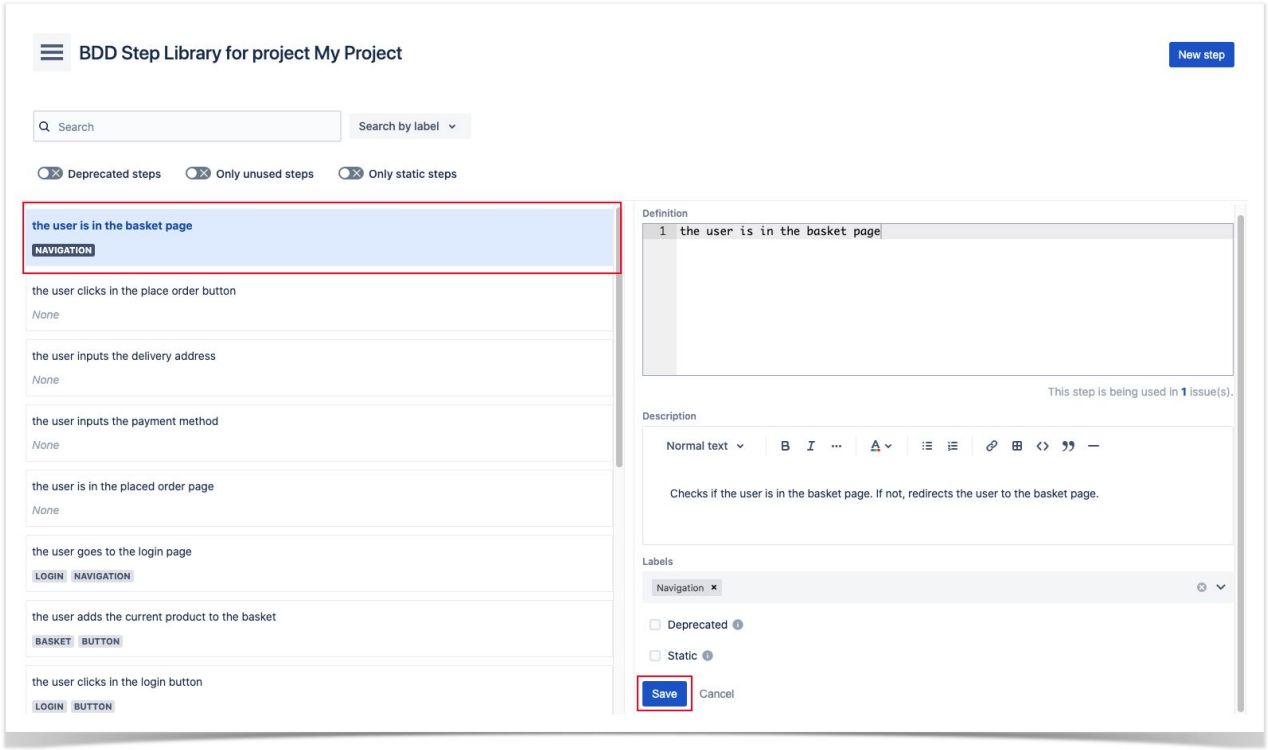
- A Test may be linked to multiple, sortable, Precondition issues that complement the test specification, that will be available during execution.
- It's something that needs to happen or be assured before the test itself. For example: "Login with admin credentials."

How to create a Precondition?

- A Precondition may be created from the "create" action
- An existing Precondition may be associated to Tests from either the Test or the Precondition issue screen

BDD Step Library (Cucumber Test)

When creating Cucumber Tests, you'll likely find it useful to re-use existing test steps and Preconditions. You can see all of the gherkin steps and edit them in the BBD Step Library.



BDD Step Library:

- Contains all of the gherkin steps used by all tests/preconditions
- Provides a overview of all automated (gherkin) steps
- Easily re-use, manage and edit steps

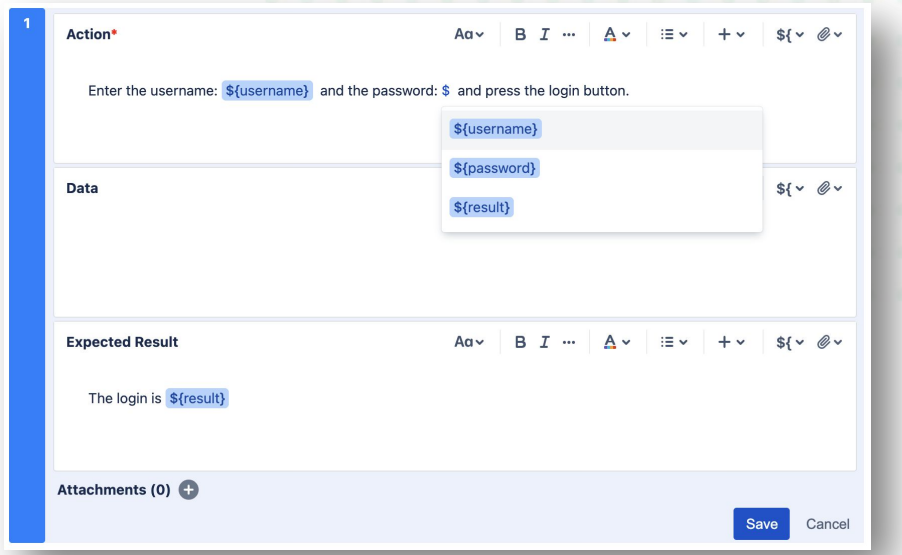
BDD step library can be managed in the Testing Board. Learn more about the BBD Step Library at [Xray Documentation]

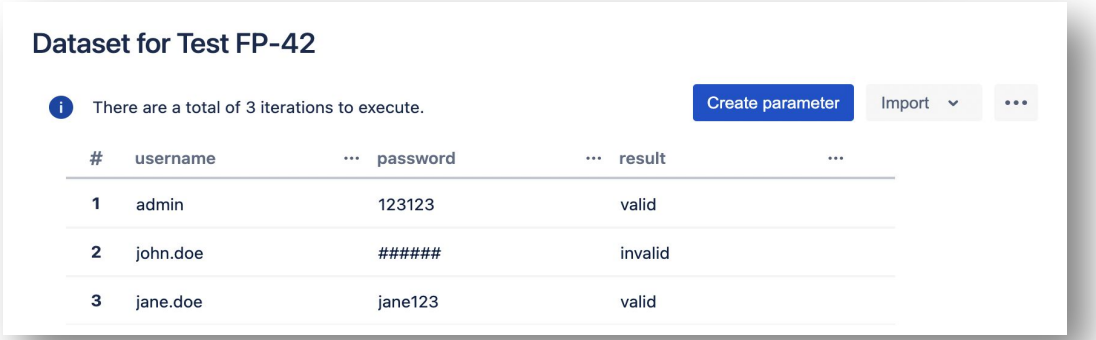
Working with Parameterized Tests

Test parameterization is a powerful practice that allows the same test to be executed multiple times with different parameters. Parameters are similar to input values (variables) that can change with each execution.

When creating a Manual Test or Preconditions, you can create steps with parameters. To manage the values of these parameters you will create Data sets.

This allows you to use the **Data Driven Testing** method.





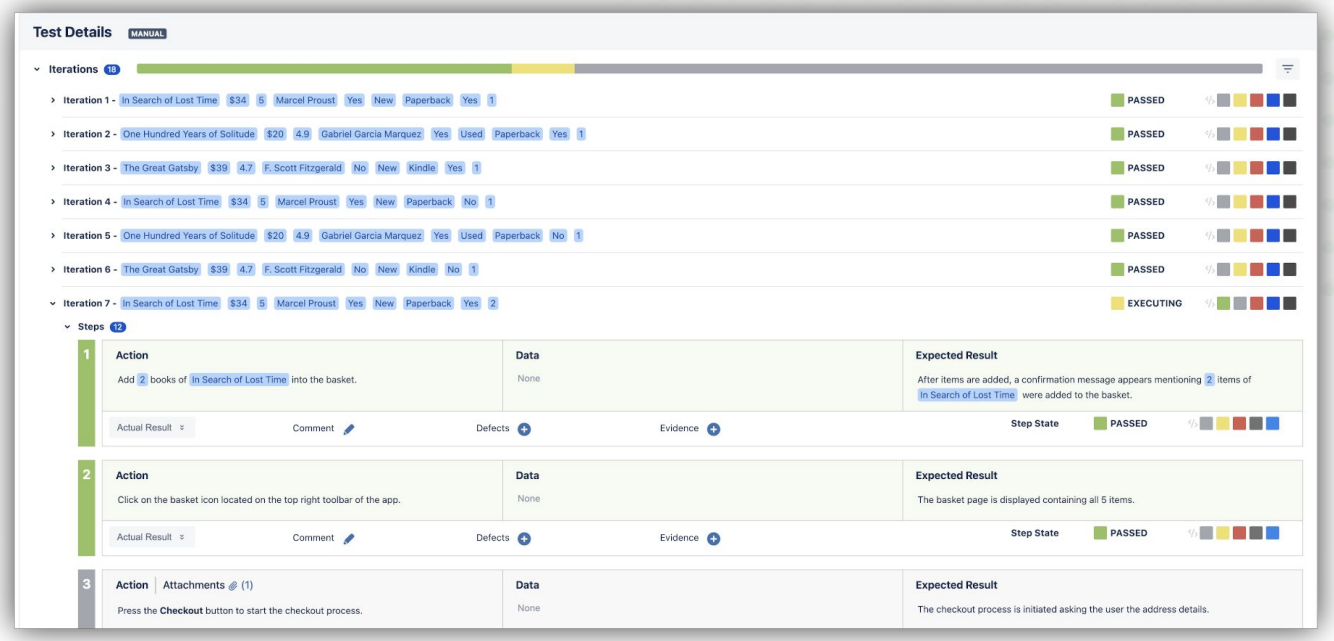
Parameters allows you to:

- Use only one test script
- Run your test script multiple times based on a data set

Parameterized Tests

A dataset can be defined in the following entities/scopes:

1. Test (default dataset)
2. Test Plan - Test
3. Test Execution - Test (Test Run)



All iterations for a given test are executed within the context of the same test run.

Each iteration can be expanded, and the steps executed individually. The step parameters will be replaced by the corresponding iteration values. The steps affect the iteration status that, in turn, affects the overall test run status.

Organizing Tests

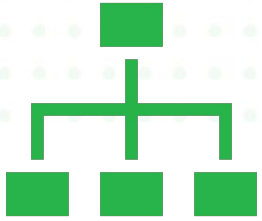
Ways to organize your Tests

Jira Way

Using Labels & fields
(Priority, Component)

Test Sets

Organize in Lists
with Test Sets



Test Repository

Organize in folders
(hierarchical
structure)

If your project has a high number of Tests, then the Test Repository may be a better fit; curiously, it can work well with newcomers and small teams as it as more friendly UI.

Test Sets works well on teams keen on Jira.

The Jira Way (labels & fields) is simpler but become harder to manage

Organizing Tests – Test Sets

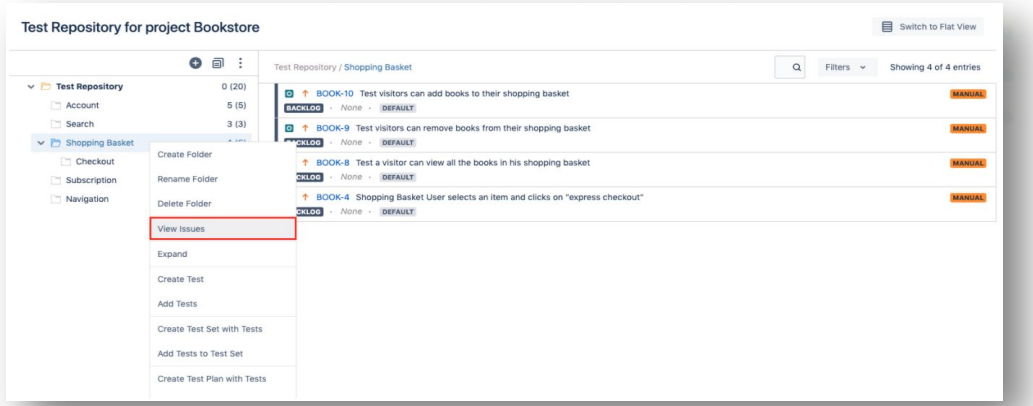
Organize Tests in flat, ordered lists using Test Sets

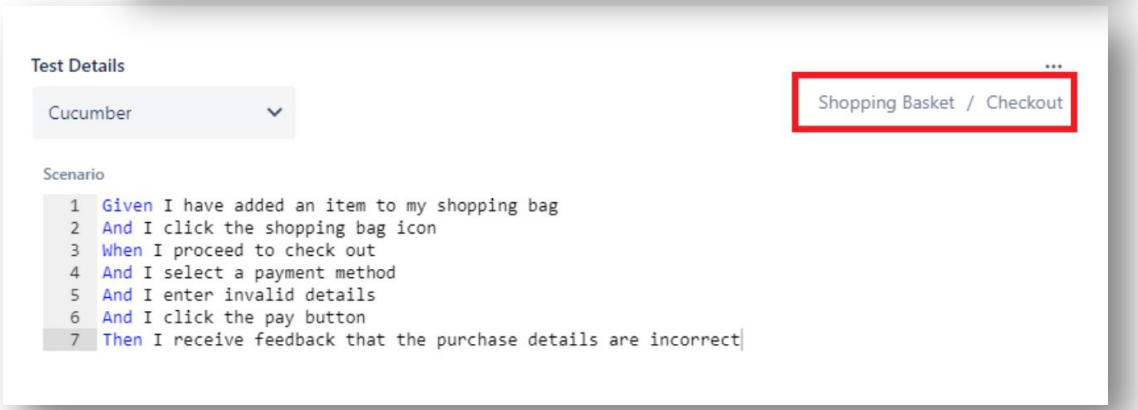
- A Test Set may be created from the “Create” action
- Tests may be added to an existing Test Set
- Tests are sortable within the Test Set

Organizing Tests – Test Repository

Organize Tests, per project, within folders within the Test Repository

- Tests may be organized by high-level feature, characteristic, component, sub-component
- Folders, or subset of Tests, can be used to create Test Sets, Test Executions, Test Plans
- A Test's path shows the location of the test within the Test Repository





Test Repository vs Test Set

Pros

- Hierarchical concept, like computer folders, may be easier to understand
- Scales better
- Can live side by side with the existing Test Set concept

Cons

- A Test can only by in one folder. It cannot be categorized in multiple ways simultaneously (as you can do using labels)
- A folder cannot be used to cover Requirements; Test Sets can

You do not need to choose between an option or another to organize your tests.

You can use all options at the same time taking advantage of the differences and features of each one.

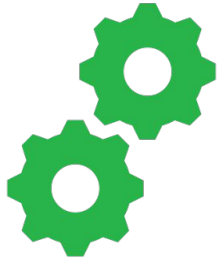
Be aware to avoid unnecessary complexities.

Planning Tests

Why plan testing?

- Planning is the activity where you decide your testing strategy.

Which requirements you want to validate?



How? Manual, Automated, Exploratory?

Whom & When? Resource Allocation & Execution Plan

Test Plan

A way to define the scope for testing and track its progress; what testing we'll be performing and its latest status.
The Test Plan is an issue that aggregates and consolidates results from multiple iterations (i.e. Test Executions).

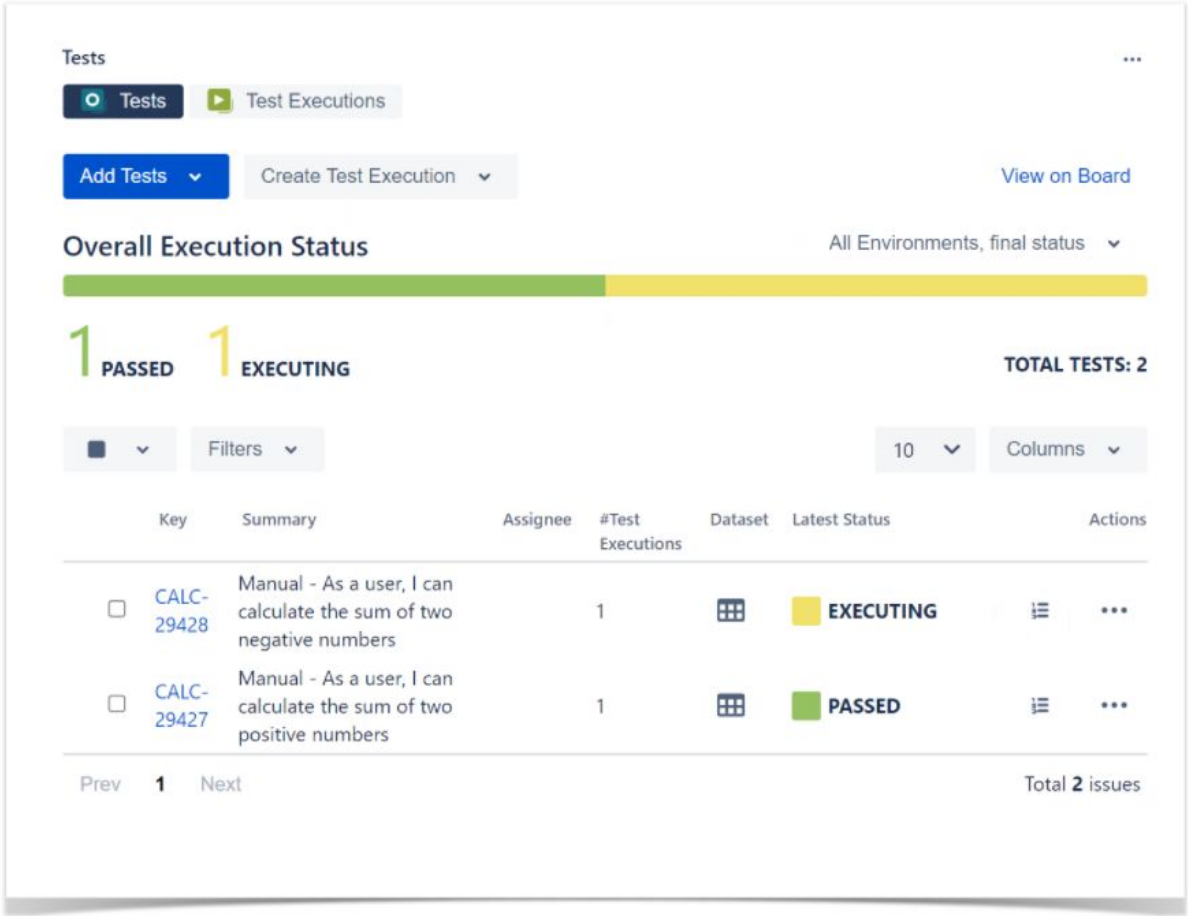
What can you do with a Test Plan?

- Define the Tests that that you want to perform and track, by adding them individually or from existing Test Sets
- Assign it explicitly to a version, using the "Fix Version" field. Related Test Executions will inherit it and affect coverage for that version
- Check the overall progress of the Tests in the Test Plan
- Create/schedule Test Executions for all planned Tests or for the ones currently on specific statuses

Test Plan

Test Plan improves visibility for what matters

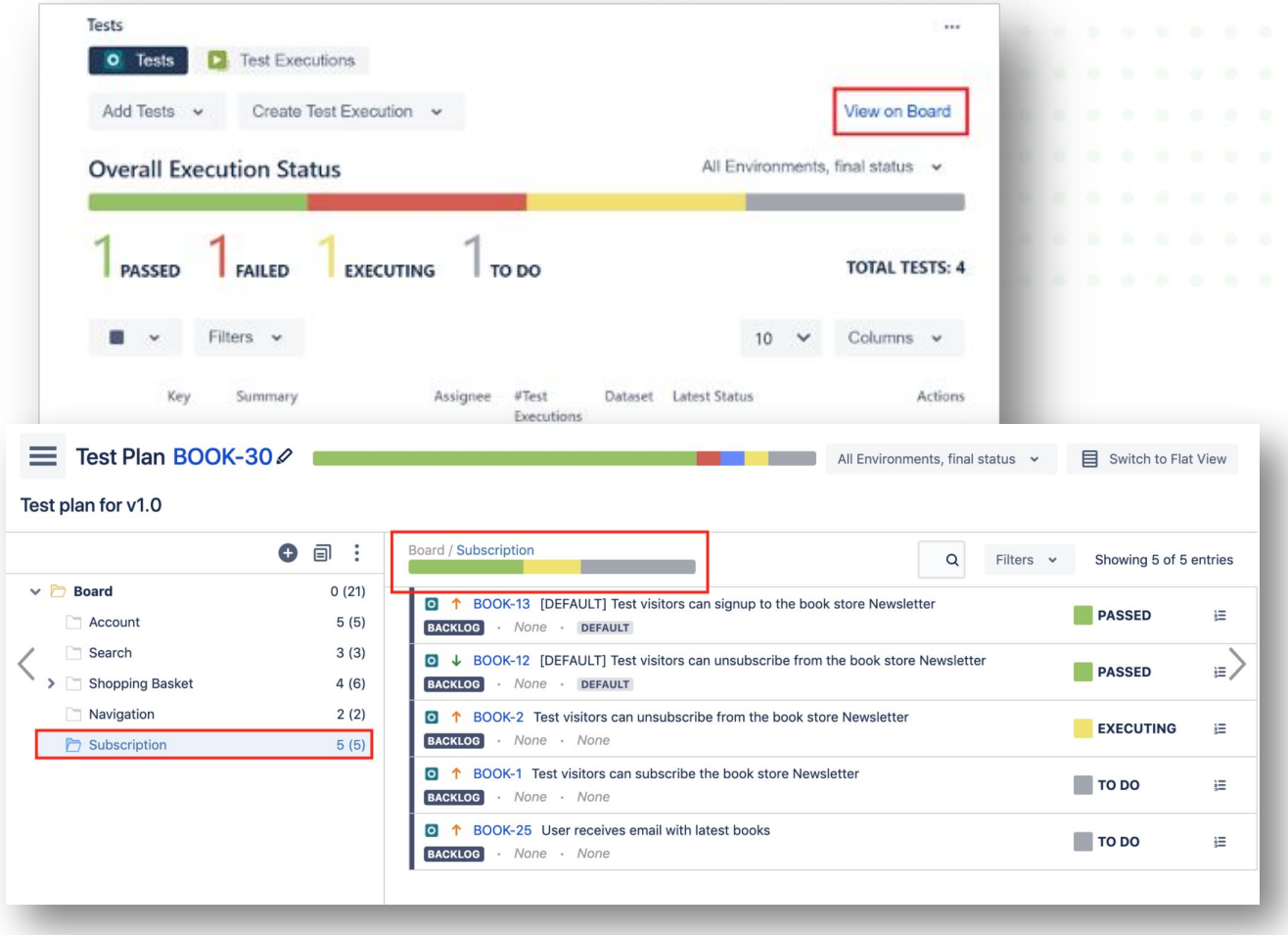
- See the status of all the Tests that belong to the Test Plan, considering their latest execution and Test Environments
- See all Test Runs performed for each Test, in the scope of this Test Plan
- Schedule executions for some/all Tests



Test Plan Board

Per Test Plan hierarchical Tests organization

- An alternate approach to using the standard, flattened Test Plan issue screen
- Can be accessed from the respective Test Plan issue screen or from project tab



Test Plan Board

What can you do with a Test Plan Board?

- Organize the Tests in a way that fits best that plan, in a folder like organization
  (by business case, by new features vs regression, etc)

- Evaluate the progress of your testing at folder level, or globally, and by:
  - Test Environment
  - jointly, for all Test Environments where you have executions on

- Filter Tests from the Test Plan and do operations over them, or over folders

- Folders, or subset of Tests within folders, can be used to create Test Sets, Test Executions

Executing Tests

Executing Tests

Test Execution

A “task” for executing a group of tests on a given version of the system, on a given environment. This task will also contain the results when they’re reported.

How to create/schedule Test Executions?

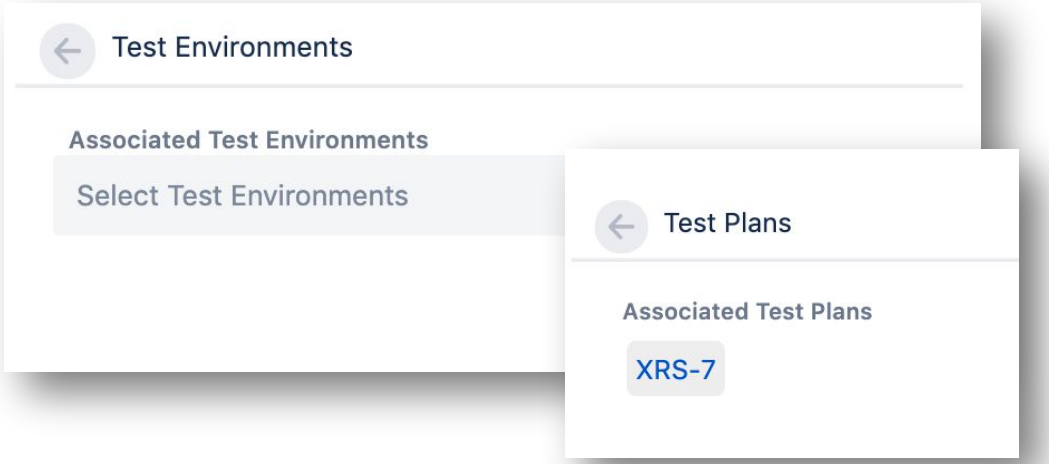
- From a
  - Test Plan (this is usually called a planned execution)
  - Test (ad hoc Test Execution)
  - Requirement as a Sub-Test Execution
- Set the “Fix Version” and, optionally, the Revision; optionally, set the planned execution date interval.
- You may also identify the Test Environment (e.g. browser) so you can analyze results and coverage on that Test Environment later

Fields available for Test Execution issues

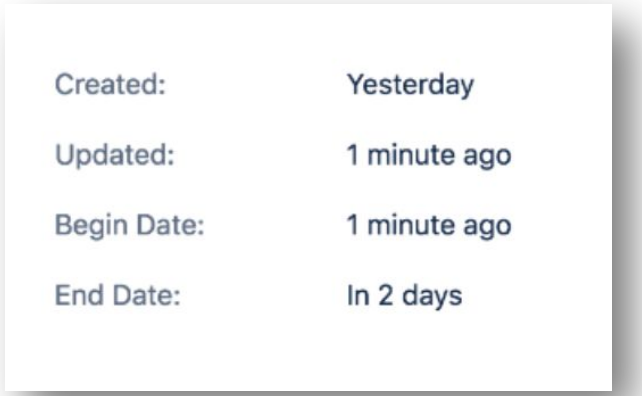
Xray provides several fields for Test Execution issues; some are standard Jira custom fields but other are internal and thus appear in specific Xray panels.

- **Begin Date** (CF) – the planned date for execution start
- **End Date** (CF) – the planned date for execution to end
- **Revision** (CF) – a complementary way to identify the SUT; a text field, usually with the source code revision(e.g. tag) or build number targeted by testing
- **Test Environments** - environments in which the Test Run is going to be/was executed
- **Test Plans** – associate and report the results of this Test Execution to some Test Plan(s)

Xray panels with internal Xray “fields”



CFs available at Dates Section



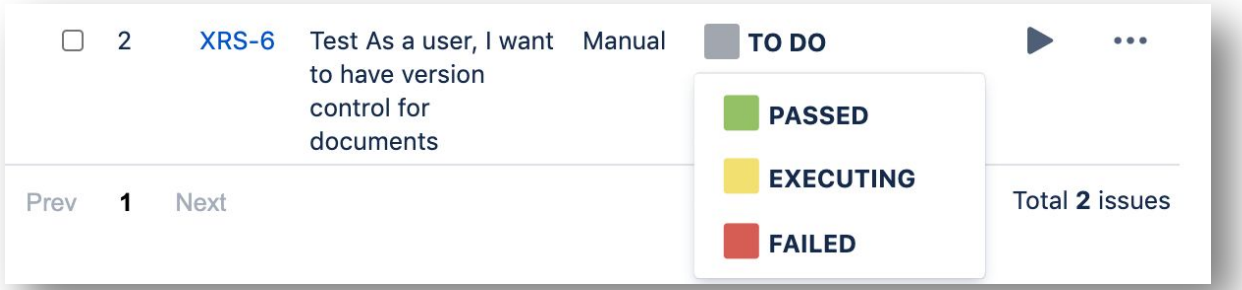
Configure the Test Execution screens, if you want

The custom fields (Begin Date, End Date, and Revision) are automatically created by Xray. However, if you want to make them visible in the screens, you need to add them by yourself.

Tests section in Test Execution issue screen

What can we do in the Tests section?

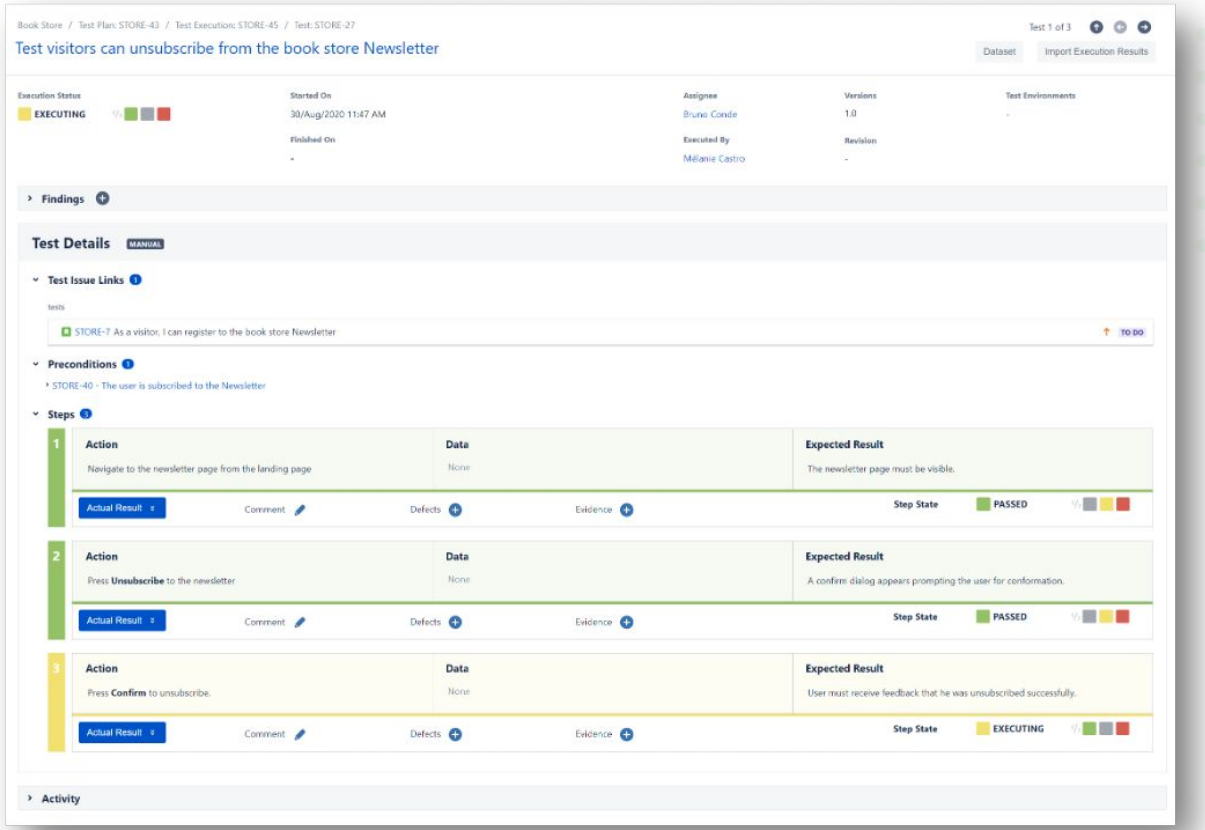
- Check the testing progress using the Overall Execution Status
- Add Tests to be run (creates an assignable Test Run per each Test)
- Reorder & assign Test Rus
- Execute a Test Run (inline or within Execution Screen)
- Archive Test Run (hide for issue views, reports, and coverage calculation)



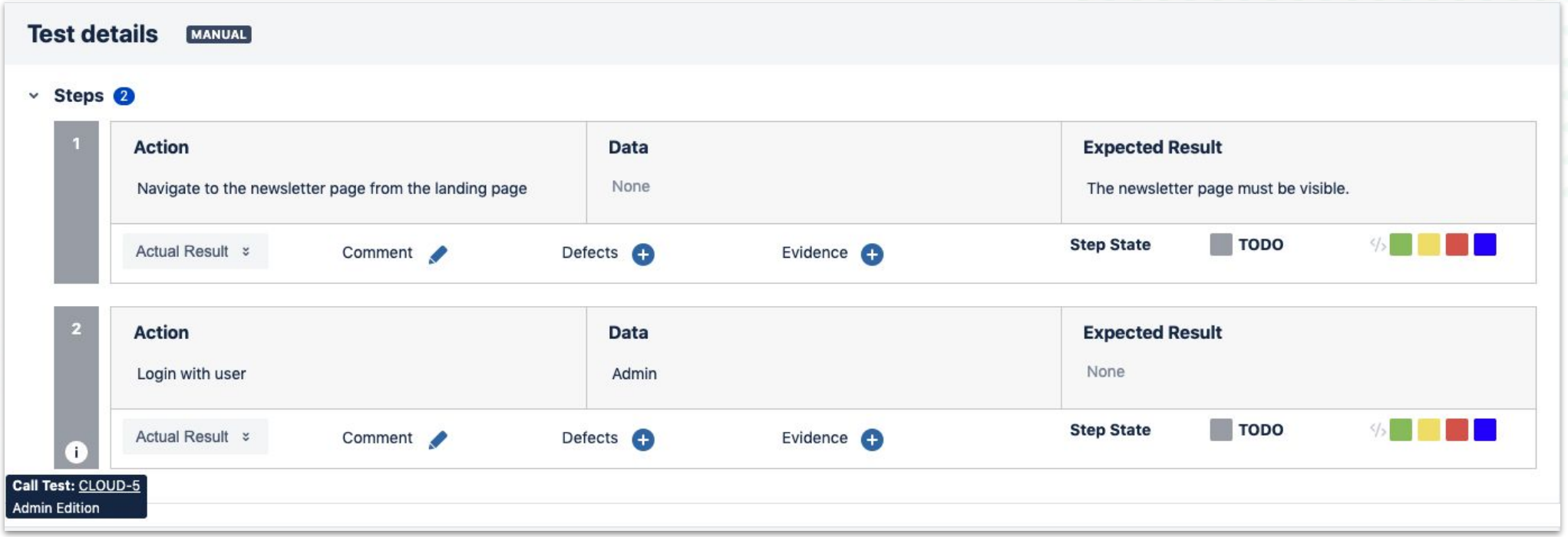
Execute Manual Tests (i.e. test cases)

Execution Screen of a Test Run (Manual Test)

- A Test Run contains a copy of the Test specification; if it changes, an alert is shown providing some actions (reset or merge)
- Comments, evidences and defects can be created at the Test or Step level
- Defects may be created in a step and the description will contain the previous steps in order to know how to reproduce the bug
- Steps may be executed out-of-order
- Changing a step status will reflect on the Test Run status



Execute Manual Tests with called Tests



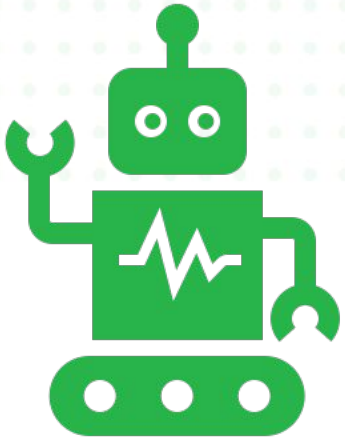
When executing tests composed of modular tests, Xray will unfold/expand all steps and replace the parameters with their resolved values.

You can check the Called Test reference at the Information button **(i)**.

Execute Automated Tests

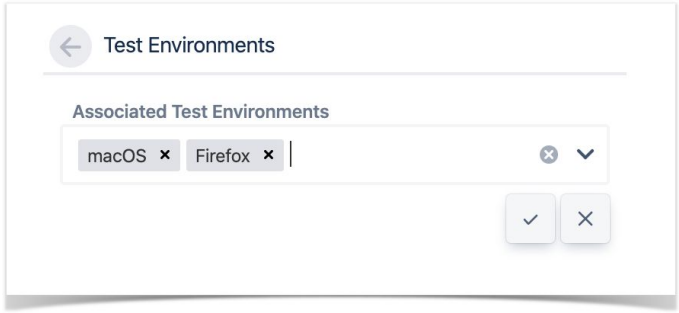
Automated Tests (Cucumber or other Generic Tests)

- Results normally are imported using the REST API
- Results can be imported by the UI
- Results can be overridden in the UI
- Execution is not done/triggered in Xray, even though you can use Jira Cloud's Automation feature to accomplish it

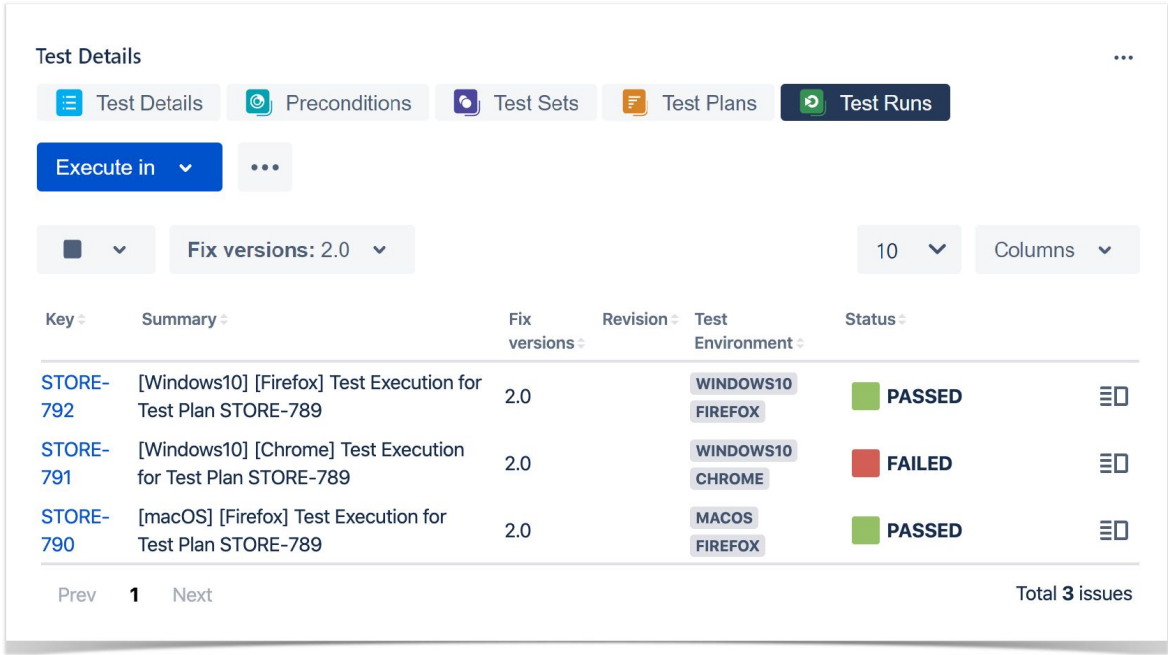


Test Environments

A Test Environment identifies where testing occurred. It can be used to identify the browser, mobile device, operating system, or even a testing stage (e.g. staging, preprod).



- A Test Execution is usually assigned to one Test Environment
- A given Test may be executed on different environments and have distinct results
- The status of a Test in environment X depends on the latest result for that environment X
- The overall status of the Test considers the latest execution on the different executed environments



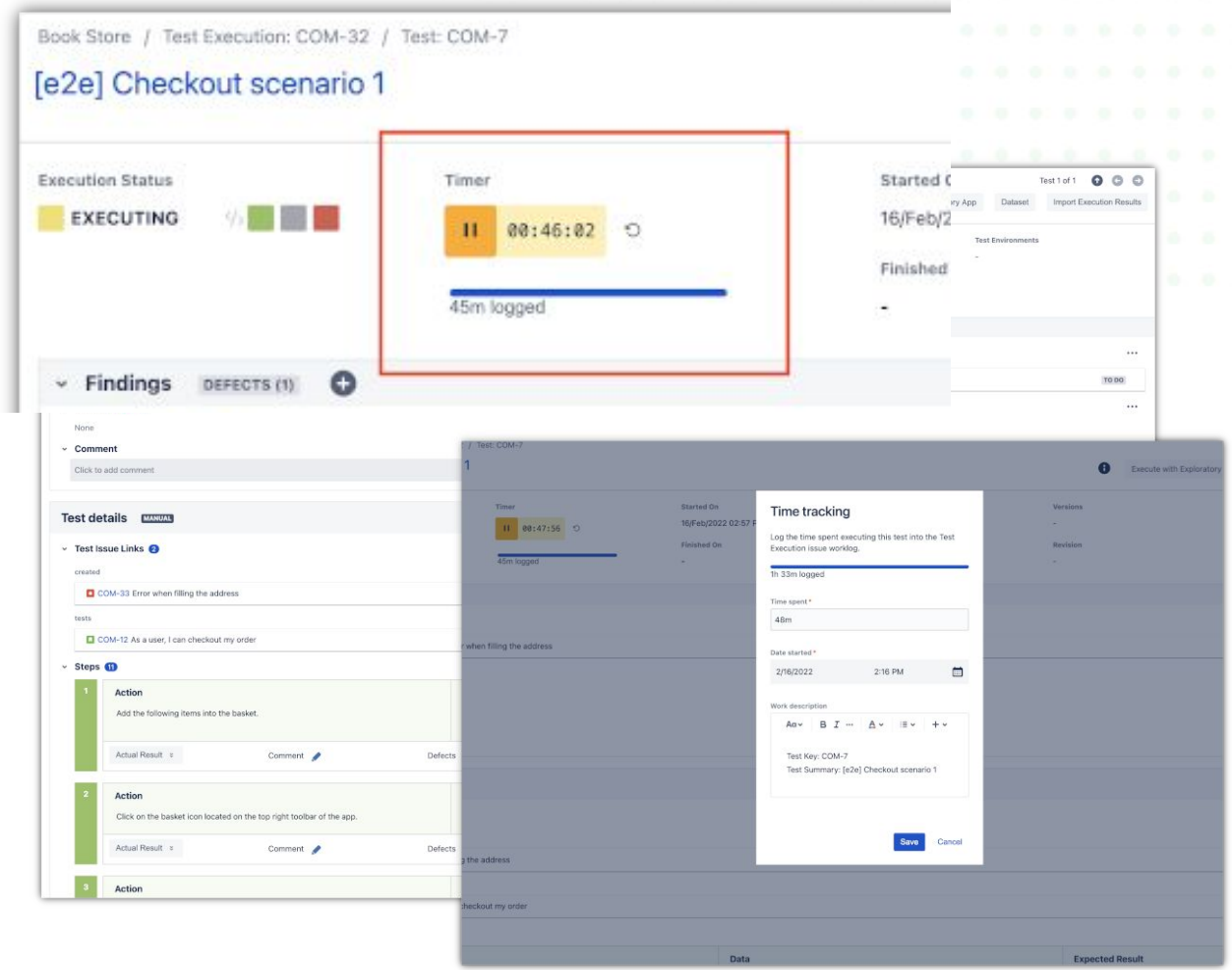
Time tracking - Timer

You can record the time spent executing a test by controlling a stopwatch (Timer).

The stopwatch is also **synchronized with the Test Run status**:

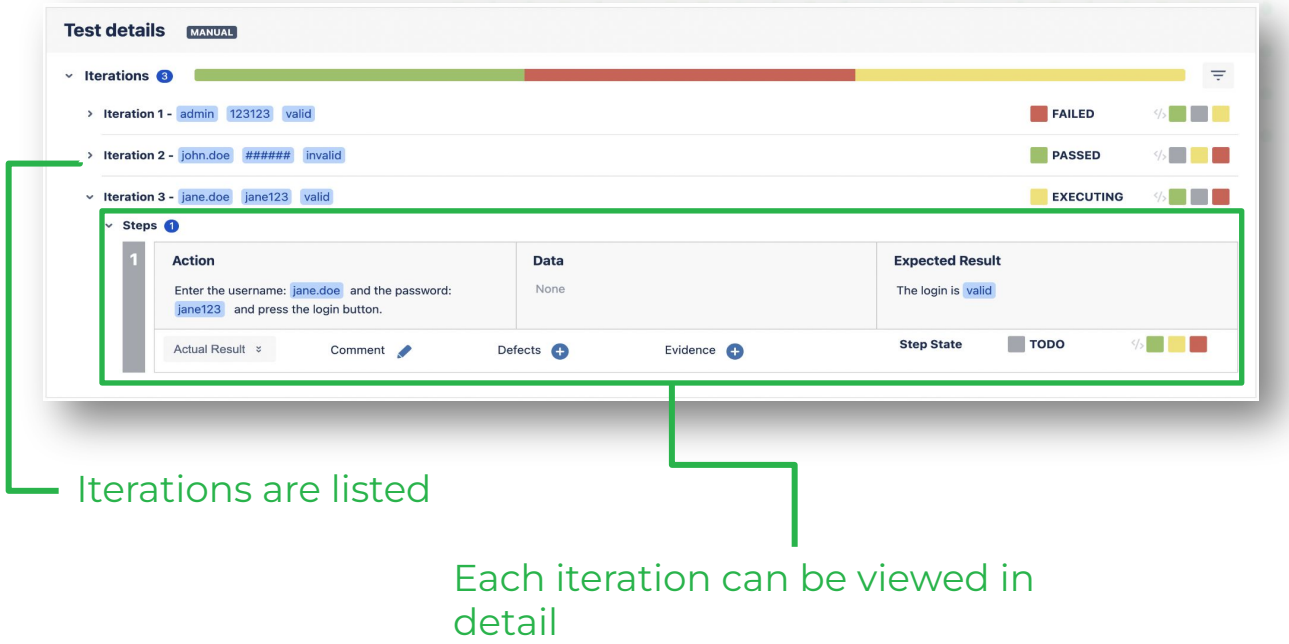
- When you start executing a step, the stopwatch starts counting.
- When you set a final status on the Test Run, the stopwatch stops.

You can also log work directly from the execution screen, just like you do within Jira issues. Any logged time will be added automatically to the Test Execution issue log work.



Execute Parameterized Tests

- When executing Tests with parameters, Test Executions are automatically created with the possible iterations
- An iteration is a sub-version of the Test with the parameters from the Data Set
- Each iteration corresponds to a line in the Data Set
- Each iteration can be run independently and have its own defects and evidences



Test Environments vs. Parameterized Tests

Although Test Environments can be thought of as a particular case of parameterized tests, datasets might not be the best choice for environmental variables that are not embedded in the test specification (e.g. manual test steps).

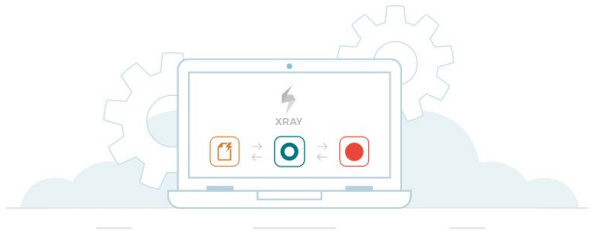
As such, it is not good practice to use datasets with environment variables such as Browsers, OSs, Databases, or Devices.

- Test engineers might want to execute the tests oriented to the environment and not the test case.
- If a test case fails only in a specific browser, once the bug is fixed, we might not want to re-execute the same test for all databases and OS's as we are confident that the change did not affect these variables. With datasets, if one iteration fails, you need to re-execute all the iterations again.
- Xray does not provide reports based on parameters.

If your parameters are environment variables that do not need to be included in the test specification, don't use datasets. Use test environments instead.

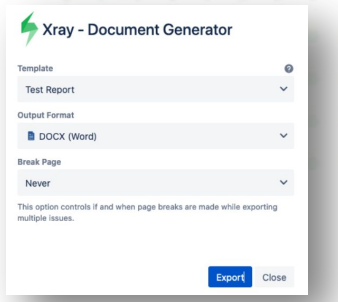
Reporting

Xray Reporting possibilities



Built-in Reports

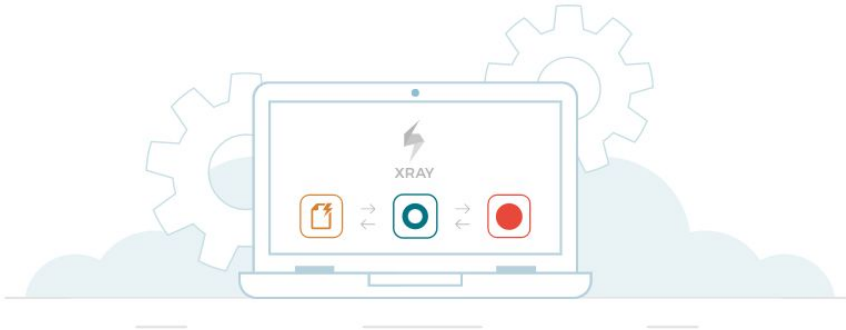
Dashboard Gadgets



Document Generator

Built-in reports

Help you see and analyze the key and essential metrics related with testing. Available at “Reports.”



Pros:

- Ready & easy to use; no configuration needed
- Available at the project context
- Essential testing metrics already setup

Cons:

- Limited number of reports
- Limited customization

Dashboard Gadgets

Display relevant testing information and metrics in a dashboard that can be customized, private or shared with team members.

Pros:

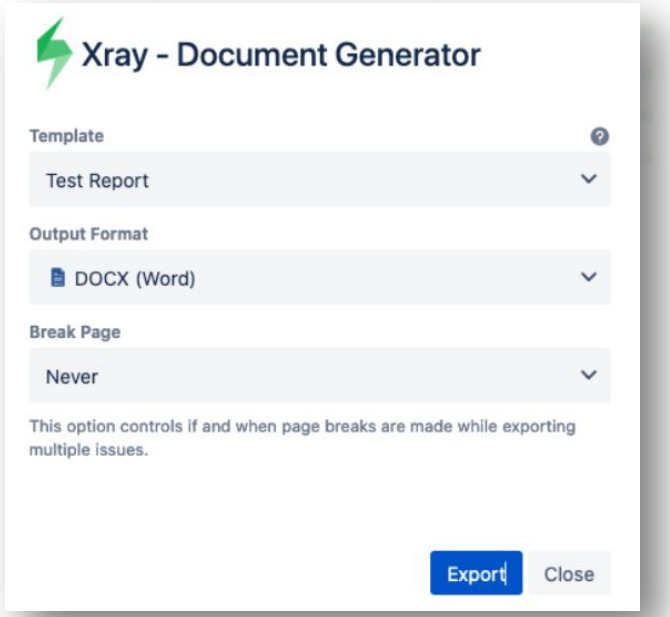
- Flexibility to create reports with predefined filters
- Possibility to analyze testing metrics alongside other relevant project metrics

Cons:

- Usually more high-level, need additional resources to drill-down information
- Changing source data means to edit gadget configuration

Document Generator

Generate reports as a document asset (e.g. doc, xls, pdf) that you can share with others (e.g. customer, manager).



Pros:

- Can be shared with stakeholders with no access to Jira
- Super customizable

Cons:

- Not real-time information (to update, you need to generate the document again)
- Does not cover BI like capabilities, so most of the reports will be based on pre-calculated format

Reporting

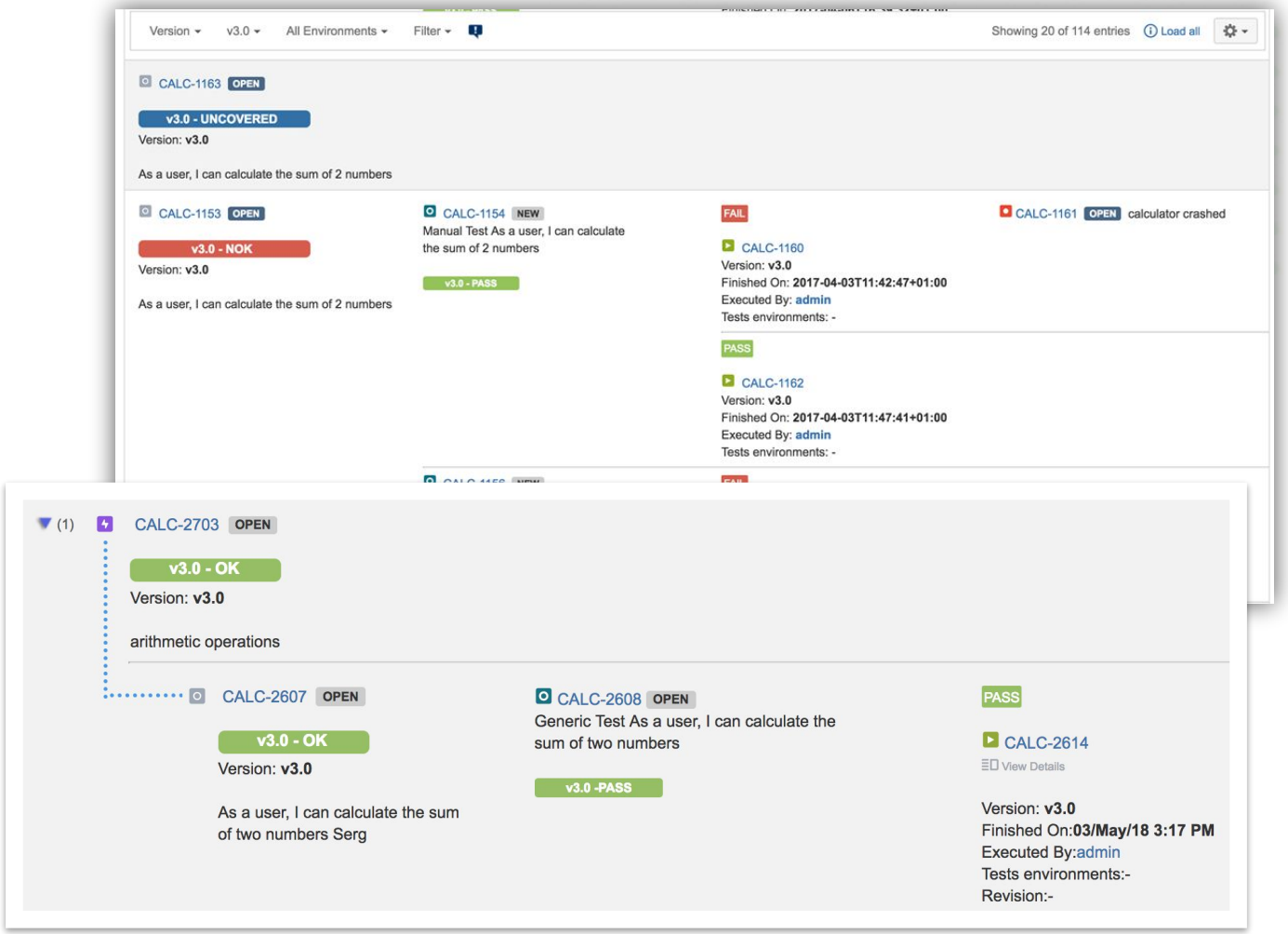
Built-in Reports

Traceability Report

Shows the requirement traceability, from requirements to defects.

Helps you analyze the requirements and related Tests, Test Runs and defects, to quickly identify uncovered or incomplete/faulty requirements.

- Can show hierarchical coverage information (e.g. Epic => Story)
- Calculate traceability for a given scope, using only the related results
- Quick filters by coverage status

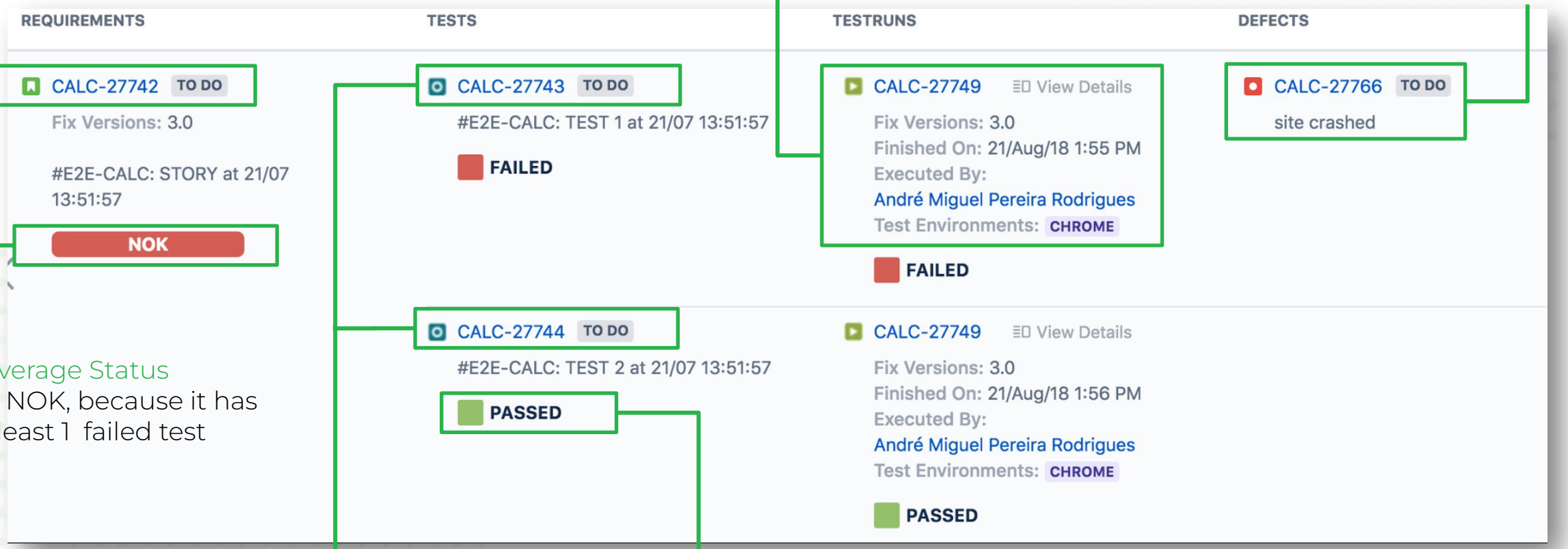


Understanding Traceability Report

Requirement Issue ID & Workflow Status
Story that should be covered by Tests

Test Run Details
Related Test Execution ID & Test Run Details

Defect Issue ID & Workflow Status
Defect found in the related Test Run



Coverage Status
It's NOK, because it has at least 1 failed test

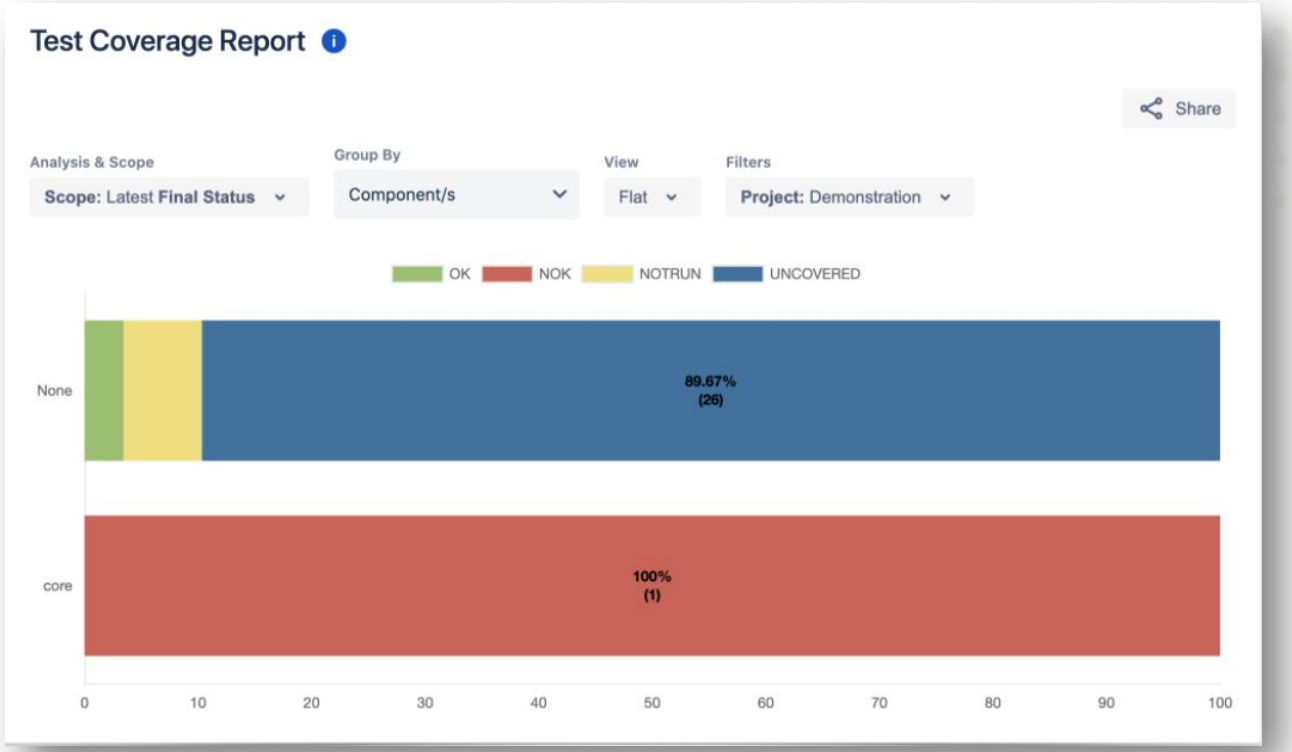
Test Issue ID & Workflow Status
Tests covering the related requirement

Test Status
It's PASSED, because the lastest Test Run is Passed

Test Coverage Report

Provides the coverage status of requirements for a given scope.

- Analysis can be done by “Version” or by “Test Plan” and on a given Test Environment
  - Takes into account results performed on that scope
- Groups requirements by component, priority, etc, so you can see the impacts on what is most important



Understanding Test Coverage Report

How to analyze the issues
Possible to analyse by latest test result; version, Test Plan and/or Test Environment.

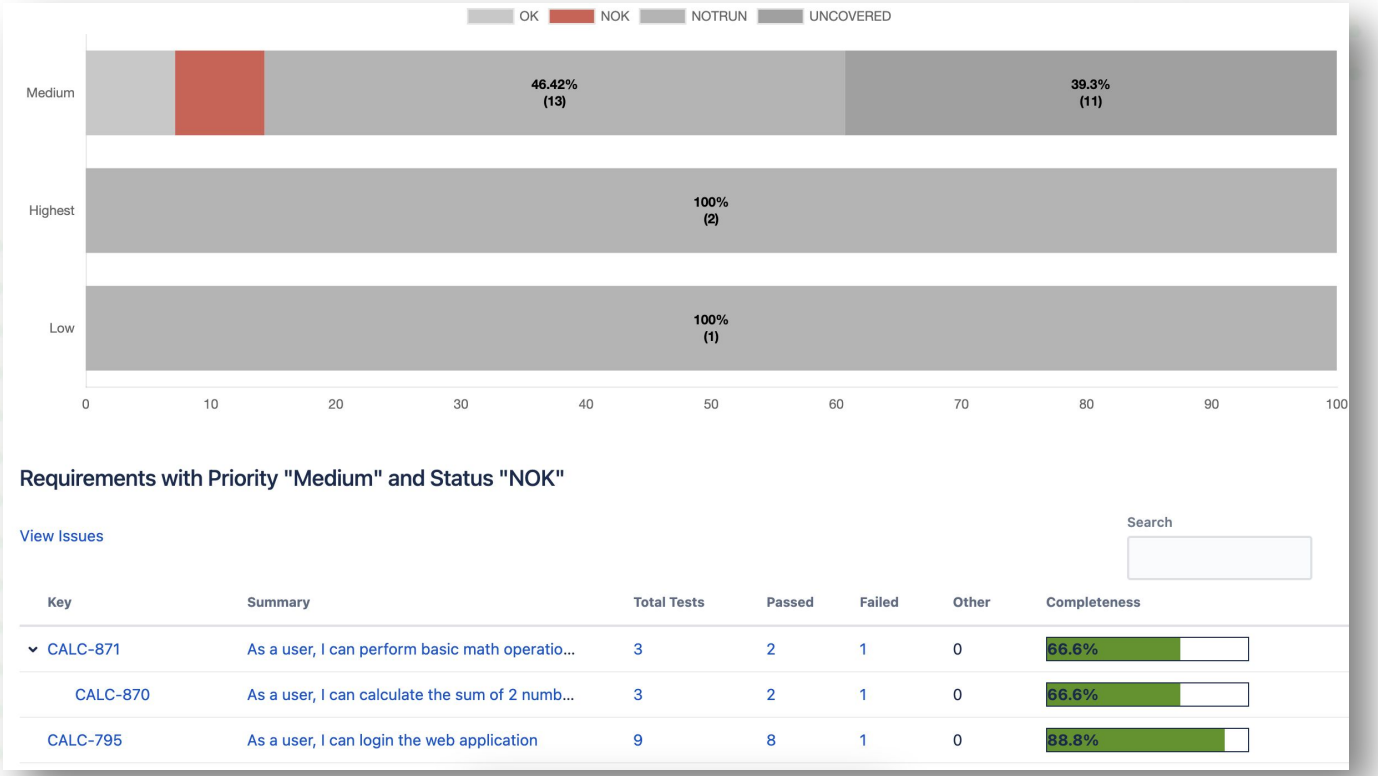
Grouping approach
How should requirements be grouped. Determined by “Group by”.

Filters
Source data (i.e. the requirements) that should be considered

Requirements in status
Percentage & number of total requirements in a given status

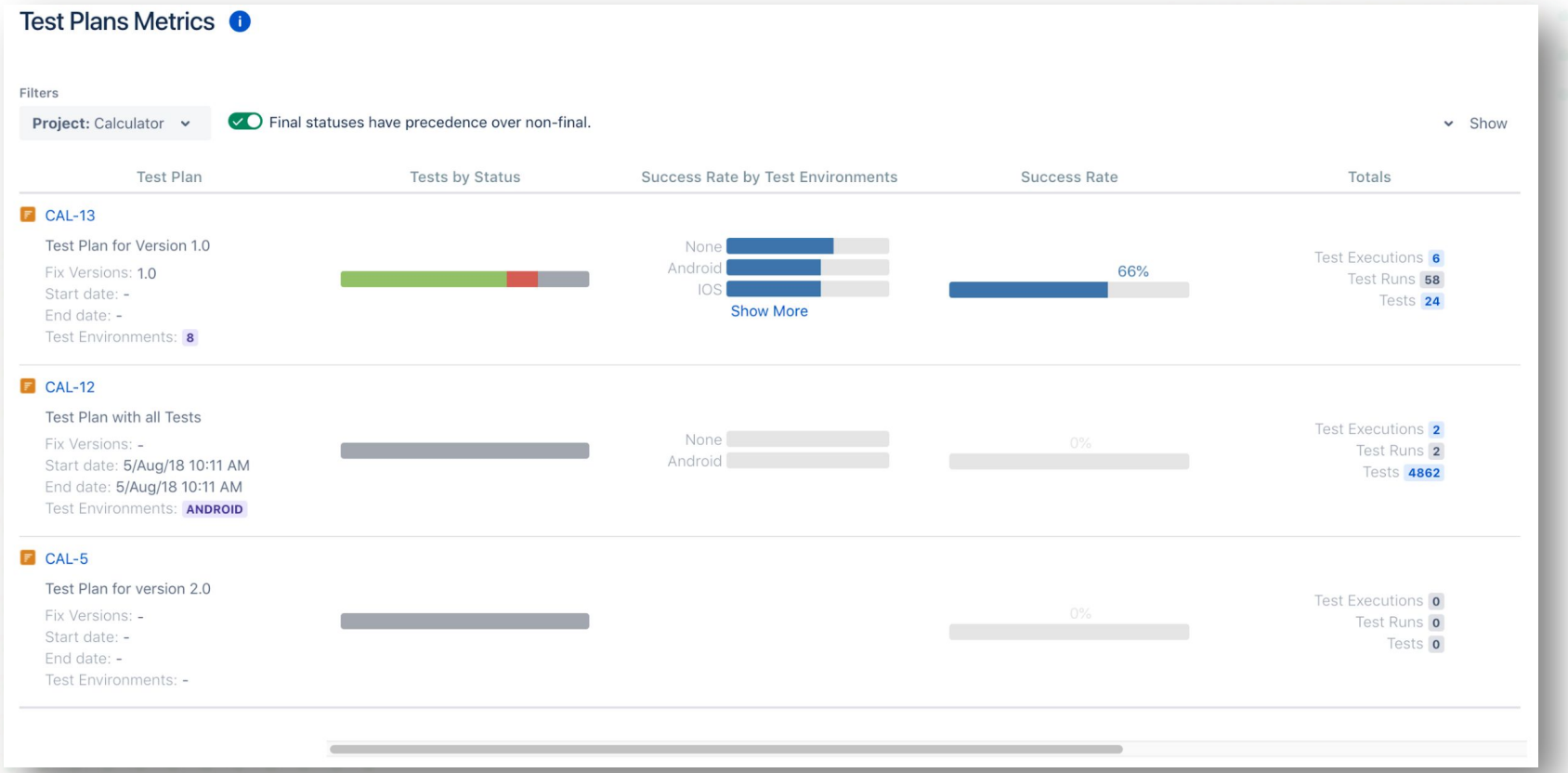
Test Coverage Report

Clicking in a bar on the chart shows the respective requirements and their completeness based on Tests passed. Sub-Requirements are also shown.



Test Plans Metrics Report

Shows a list of Test Plans with consolidated information for each one, such as Test by statuses, overall progress and Test Environments metrics.

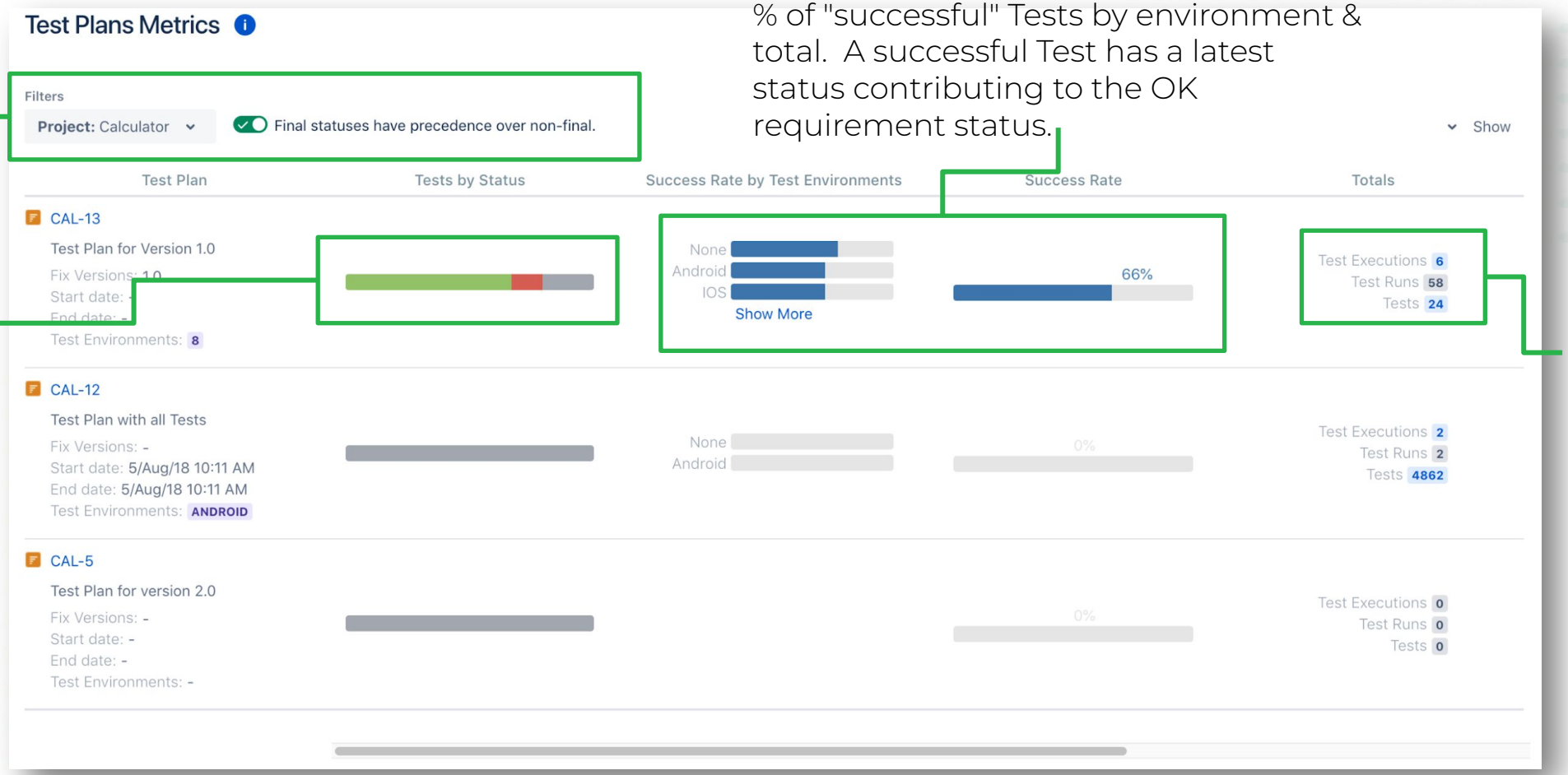


Understanding Test Plans Metrics Report

Source Data
Source data (Test Plans) that should be considered

Tests by status
Number of Test Plans Tests, per latest status

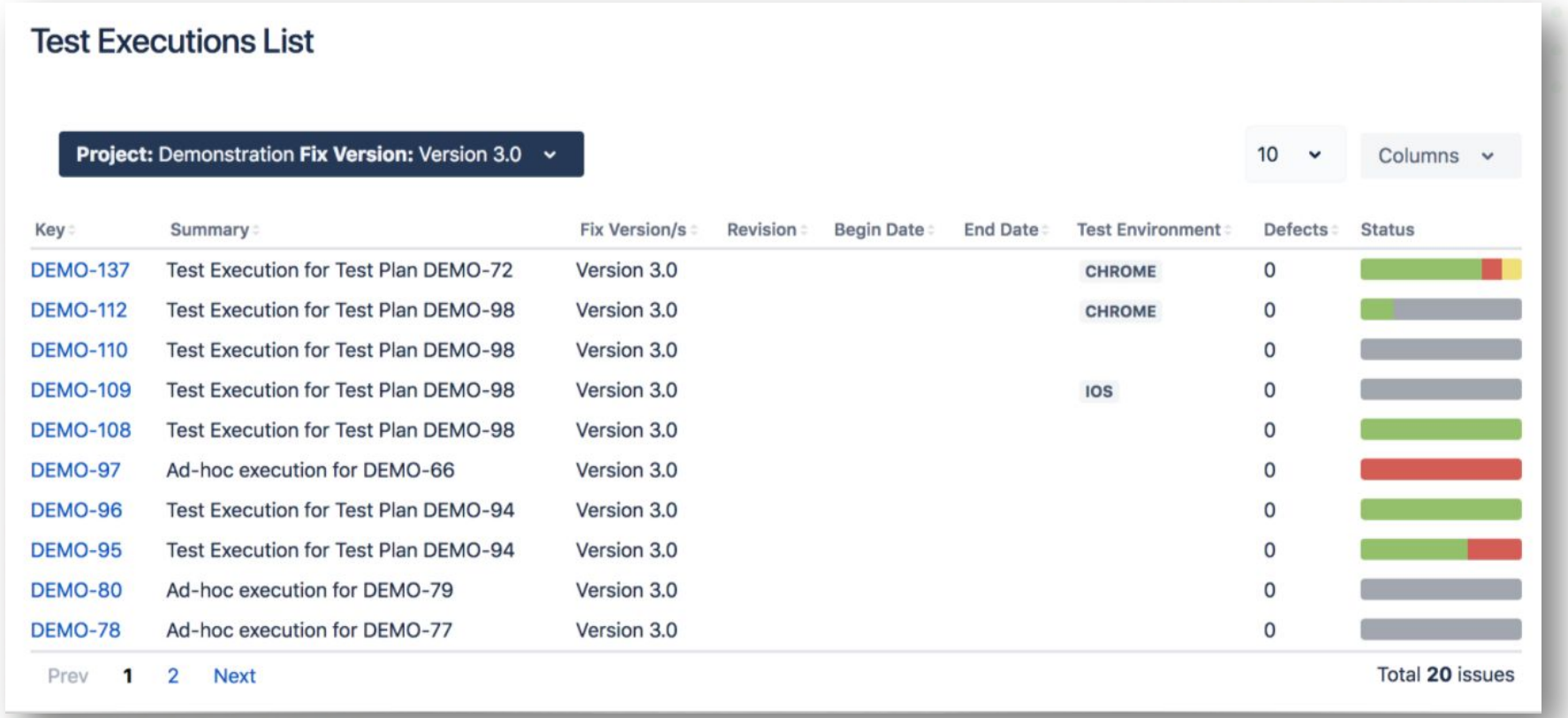
Success Rate
% of "successful" Tests by environment & total. A successful Test has a latest status contributing to the OK requirement status.



Totals metrics
Total number of Test Executions, Test Run & Tests for the related Test Plan

Test Execution List

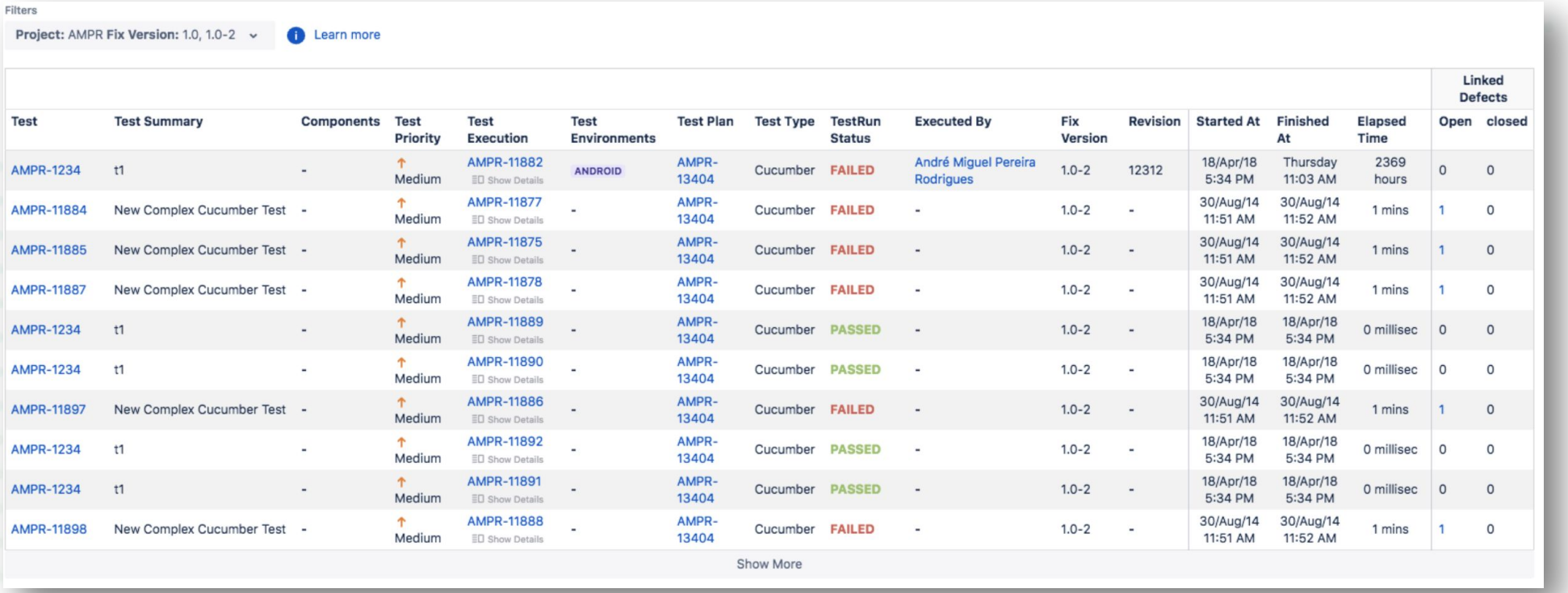
Shows a list of Test Execution issues and their corresponding execution status, including overall status and linked defects.



Test Runs List

Shows a list of Test Runs with detailed info for each one, including related Test, Test Execution and Test Plan entities, execution dates, linked defects among others.

- Filter by Test Run assignee/executed by, Test, version, Test Environment, etc



Filters

Project: AMPR Fix Version: 1.0, 1.0-2 Learn more

															Linked Defects	
Test	Test Summary	Components	Test Priority	Test Execution	Test Environments	Test Plan	Test Type	TestRun Status	Executed By	Fix Version	Revision	Started At	Finished At	Elapsed Time	Open	closed
AMPR-1234	t1	-	Medium	AMPR-11882 Show Details	ANDROID	AMPR-13404	Cucumber	FAILED	André Miguel Pereira Rodrigues	1.0-2	12312	18/Apr/18 5:34 PM	Thursday 11:03 AM	2369 hours	0	0
AMPR-11884	New Complex Cucumber Test	-	Medium	AMPR-11877 Show Details	-	AMPR-13404	Cucumber	FAILED	-	1.0-2	-	30/Aug/14 11:51 AM	30/Aug/14 11:52 AM	1 mins	1	0
AMPR-11885	New Complex Cucumber Test	-	Medium	AMPR-11875 Show Details	-	AMPR-13404	Cucumber	FAILED	-	1.0-2	-	30/Aug/14 11:51 AM	30/Aug/14 11:52 AM	1 mins	1	0
AMPR-11887	New Complex Cucumber Test	-	Medium	AMPR-11878 Show Details	-	AMPR-13404	Cucumber	FAILED	-	1.0-2	-	30/Aug/14 11:51 AM	30/Aug/14 11:52 AM	1 mins	1	0
AMPR-1234	t1	-	Medium	AMPR-11889 Show Details	-	AMPR-13404	Cucumber	PASSED	-	1.0-2	-	18/Apr/18 5:34 PM	18/Apr/18 5:34 PM	0 millisec	0	0
AMPR-1234	t1	-	Medium	AMPR-11890 Show Details	-	AMPR-13404	Cucumber	PASSED	-	1.0-2	-	18/Apr/18 5:34 PM	18/Apr/18 5:34 PM	0 millisec	0	0
AMPR-11897	New Complex Cucumber Test	-	Medium	AMPR-11886 Show Details	-	AMPR-13404	Cucumber	FAILED	-	1.0-2	-	30/Aug/14 11:51 AM	30/Aug/14 11:52 AM	1 mins	1	0
AMPR-1234	t1	-	Medium	AMPR-11892 Show Details	-	AMPR-13404	Cucumber	PASSED	-	1.0-2	-	18/Apr/18 5:34 PM	18/Apr/18 5:34 PM	0 millisec	0	0
AMPR-1234	t1	-	Medium	AMPR-11891 Show Details	-	AMPR-13404	Cucumber	PASSED	-	1.0-2	-	18/Apr/18 5:34 PM	18/Apr/18 5:34 PM	0 millisec	0	0
AMPR-11898	New Complex Cucumber Test	-	Medium	AMPR-11888 Show Details	-	AMPR-13404	Cucumber	FAILED	-	1.0-2	-	30/Aug/14 11:51 AM	30/Aug/14 11:52 AM	1 mins	1	0

Show More

Other List Reports

- Test List Report
- Test Set List Report
- Test Plan List Report

These report lists can be useful to easily display information about Xray entities. Besides displaying key information, they also display information on status.

Xray saves any filters applied to our Built-in Reports. This means that the filters will not be reset when you leave the report page and that you can always re-run the same report. You can change/update the filter when needed.

Reporting

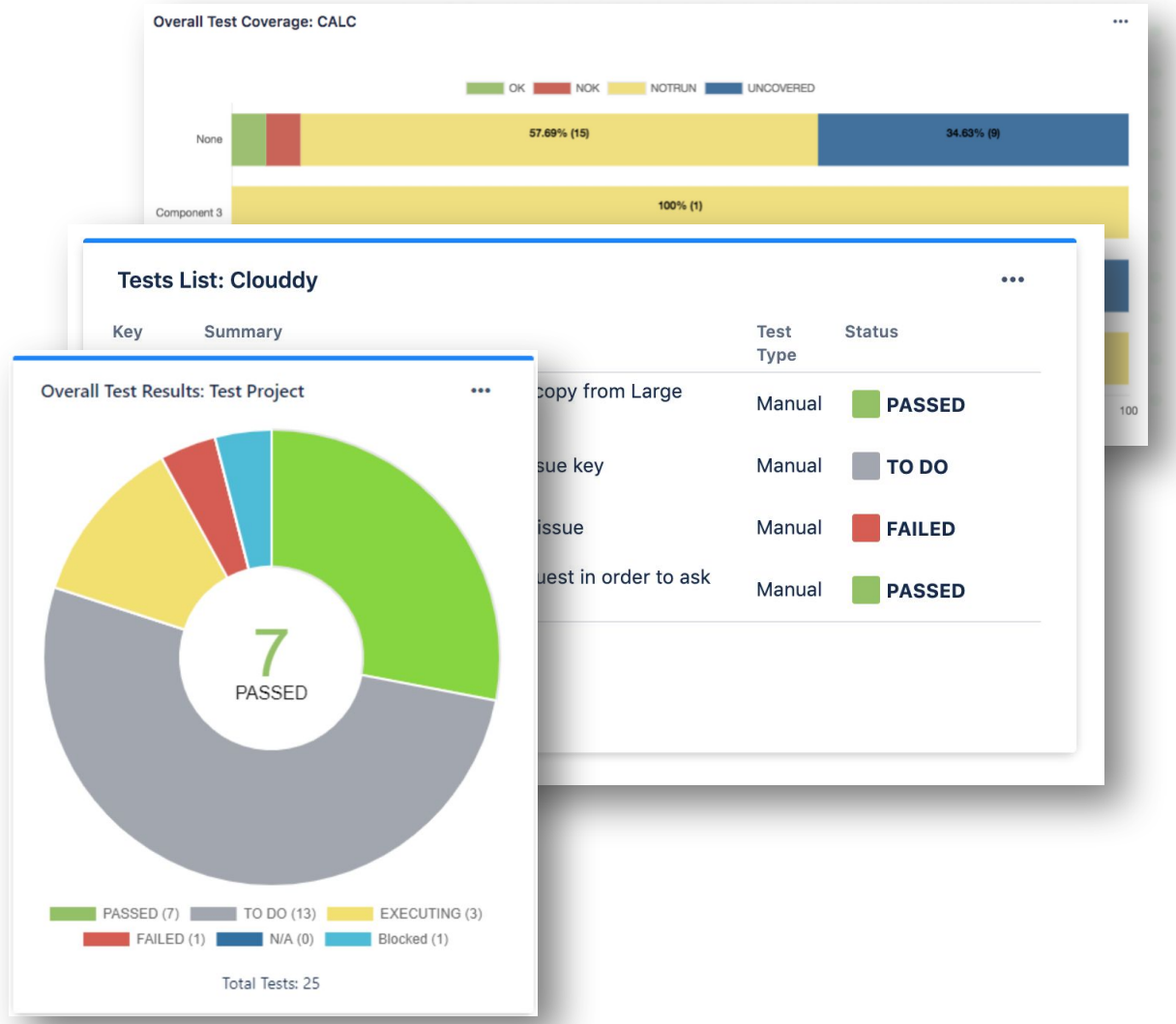
Gadgets in dashboards

Dashboards & gadgets

Dashboards are designed to display gadgets that help you organize your projects, tasks, and results in different charts.

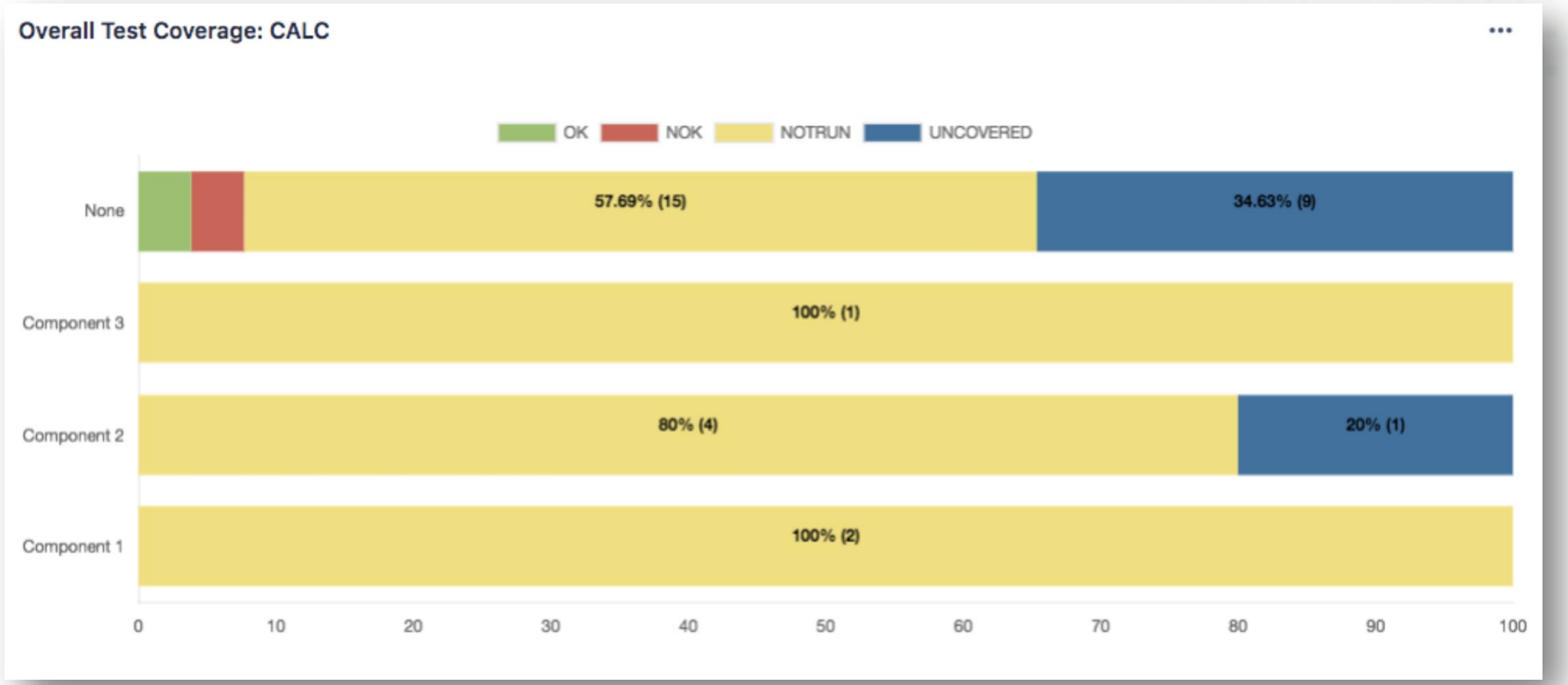
You can create new ones (private or shared).

Xray provides several gadgets that will help you keep track of your testing information.



Overall Test Coverage Gadgets

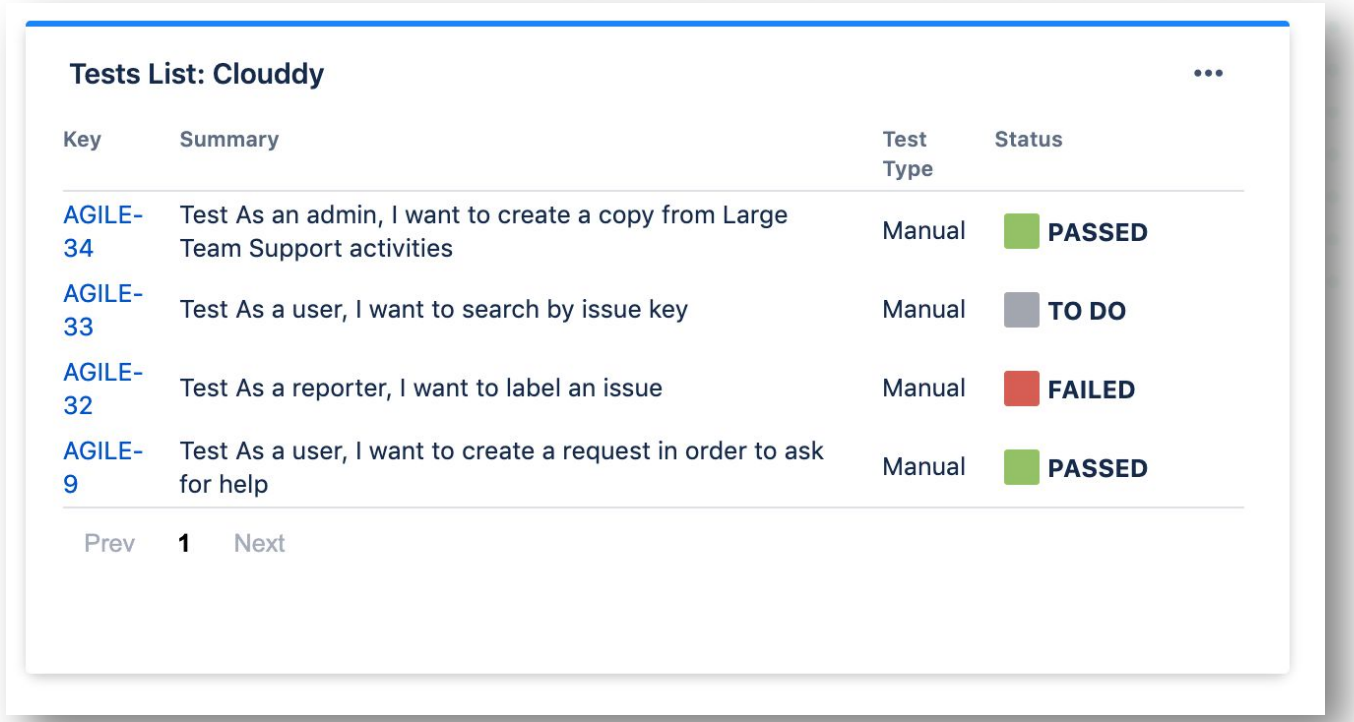
Provides a quick way of evaluating the current (coverage) status of your project's requirements. Similar to Test Coverage Report.



Tests List Gadget

Provides a quick way of evaluating the current consolidated (coverage) status for your Tests on a specific scope (i.e. Version/Test Plan).

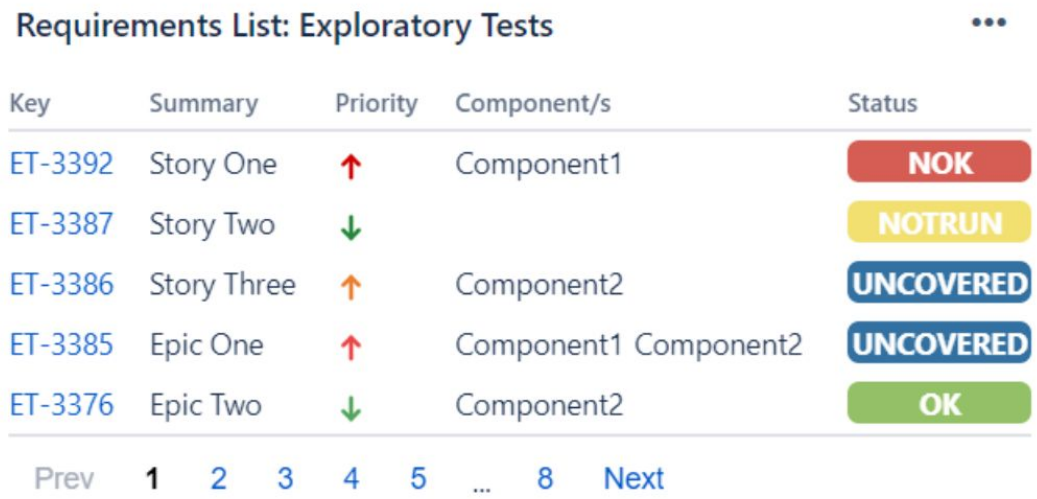
- Quickly analyze how some specific Tests are doing in some given context
- Ability to analyze the coverage per Test Environment



Requirements List Gadget

Evaluate the coverage and completeness of requirements on a specific Version/Test Plan.

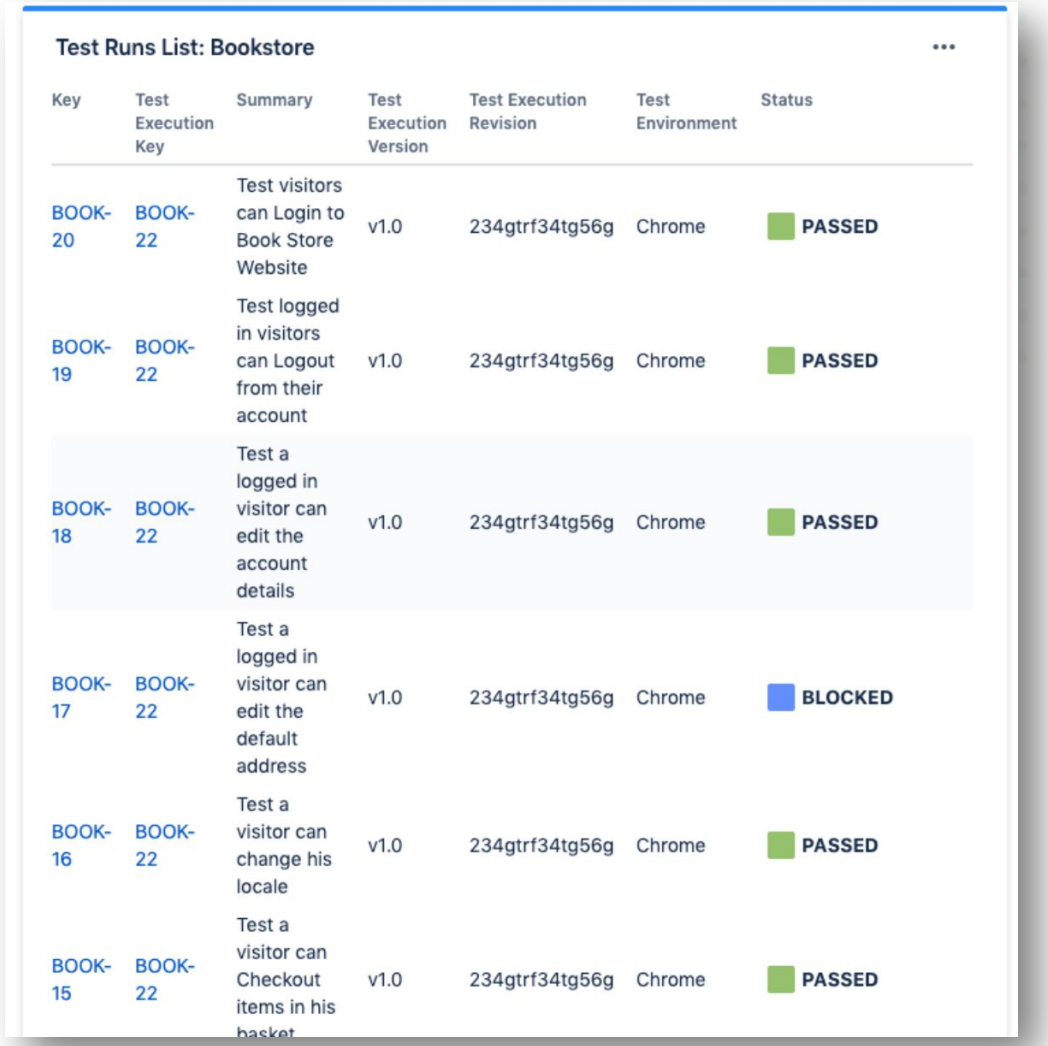
- Takes into account the results performed on that scope
- Track coverage globally or per Test Environment



Test Runs List Gadget

Shows a list of Test Runs of some given Test Executions with brief summary information, including the result.

- Filter by Test Run assignee/executed by, dates
- Ability to jump to execution screen



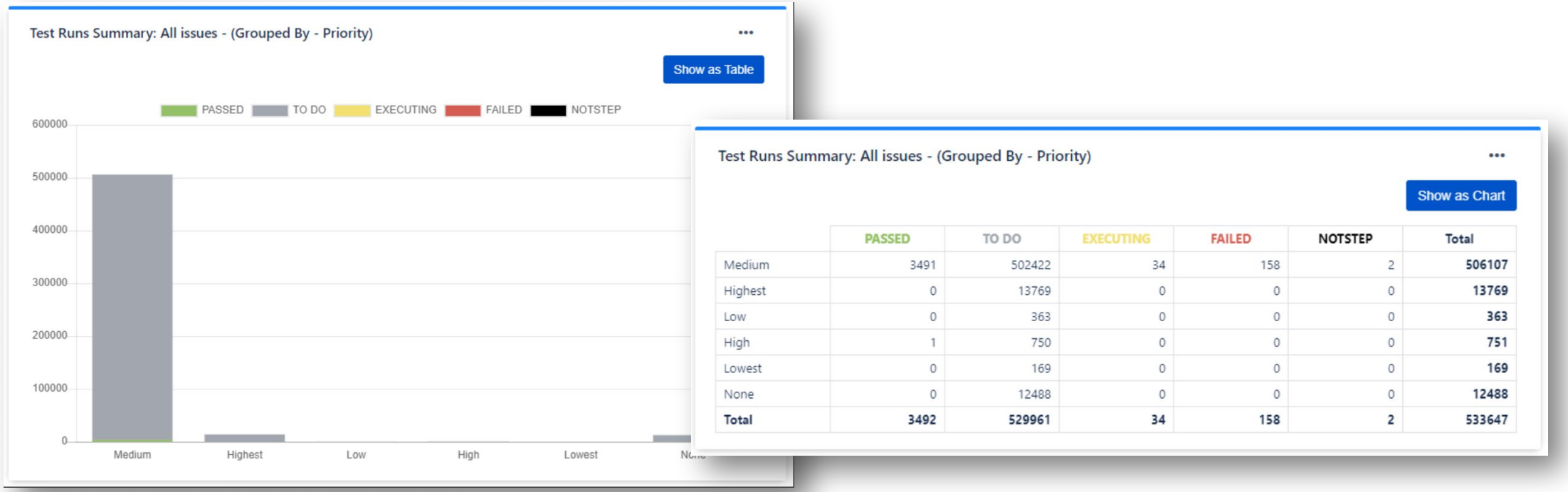
Test Runs List: Bookstore

Key	Test Execution Key	Summary	Test Execution Version	Test Execution Revision	Test Environment	Status
BOOK-20	BOOK-22	Test visitors can Login to Book Store Website	v1.0	234gtrf34tg56g	Chrome	PASSED
BOOK-19	BOOK-22	Test logged in visitors can Logout from their account	v1.0	234gtrf34tg56g	Chrome	PASSED
BOOK-18	BOOK-22	Test a logged in visitor can edit the account details	v1.0	234gtrf34tg56g	Chrome	PASSED
BOOK-17	BOOK-22	Test a logged in visitor can edit the default address	v1.0	234gtrf34tg56g	Chrome	BLOCKED
BOOK-16	BOOK-22	Test a visitor can change his locale	v1.0	234gtrf34tg56g	Chrome	PASSED
BOOK-15	BOOK-22	Test a visitor can Checkout items in his basket	v1.0	234gtrf34tg56g	Chrome	PASSED

Test Runs Summary Gadget

See the overall status of scheduled/executed Test Runs, grouped by a given field.

- Ability to filter Test Runs by Test Run's assignee and status
- Group by some fields of the Test and of Test Run

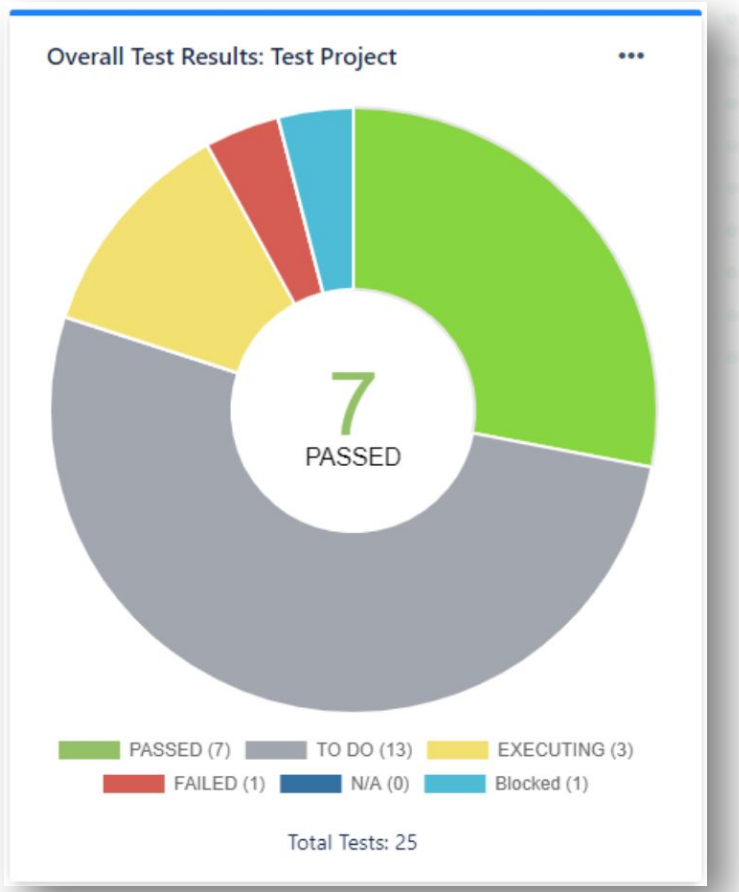


	PASSED	TO DO	EXECUTING	FAILED	NOTSTEP	Total
Medium	3491	502422	34	158	2	**506107**
Highest	0	13769	0	0	0	**13769**
Low	0	363	0	0	0	**363**
High	1	750	0	0	0	**751**
Lowest	0	169	0	0	0	**169**
None	0	12488	0	0	0	**12488**
Total	**3492**	**529961**	**34**	**158**	**2**	533647

Overall Test Results Gadget

Provides a way to analyze the overall, consolidated test results distribution by their status.

- Track number of Tests, by current status, for a specific Version/Test Plan (and optionally, on a specific Test Environment)
- Analyze the overall progress of a Test Plan



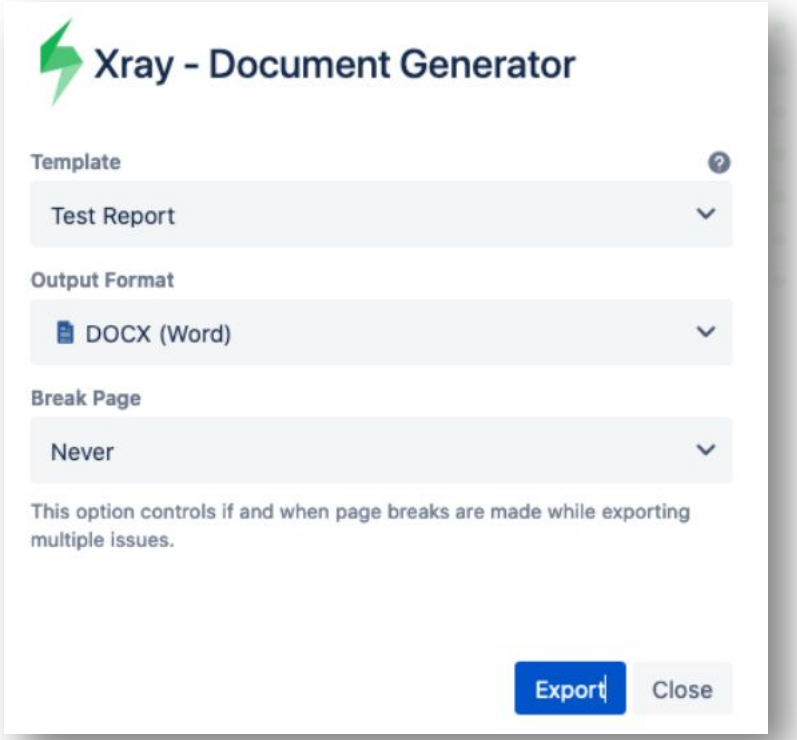
Reporting

Document Generator

What is the Document Generator?

Xray Document Generator is a feature that allows you to customize your own templates and generate documents regarding your Jira data, mainly, Xray issues, requirements, and defects.

It's easy to set up and start creating your documents.

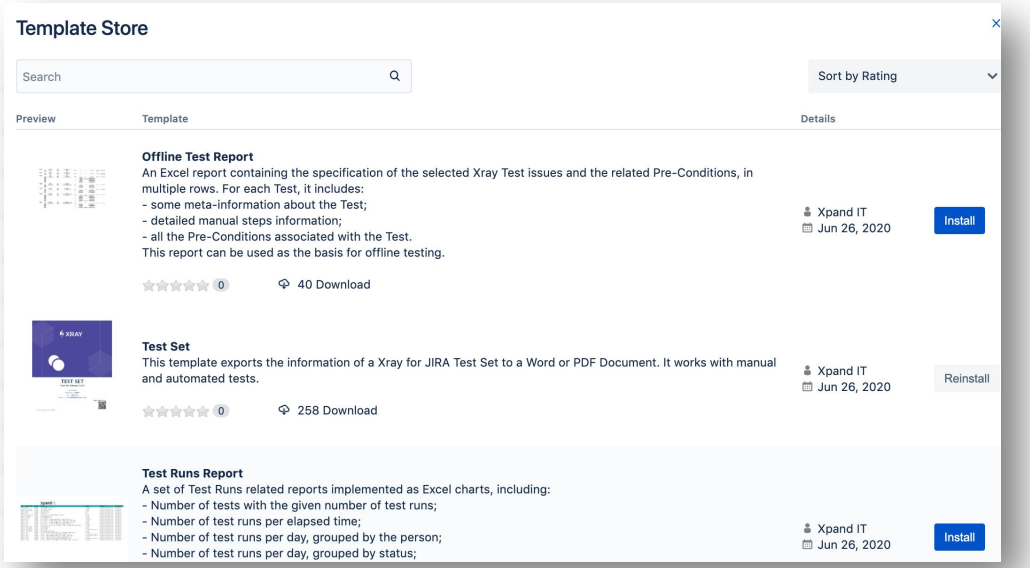


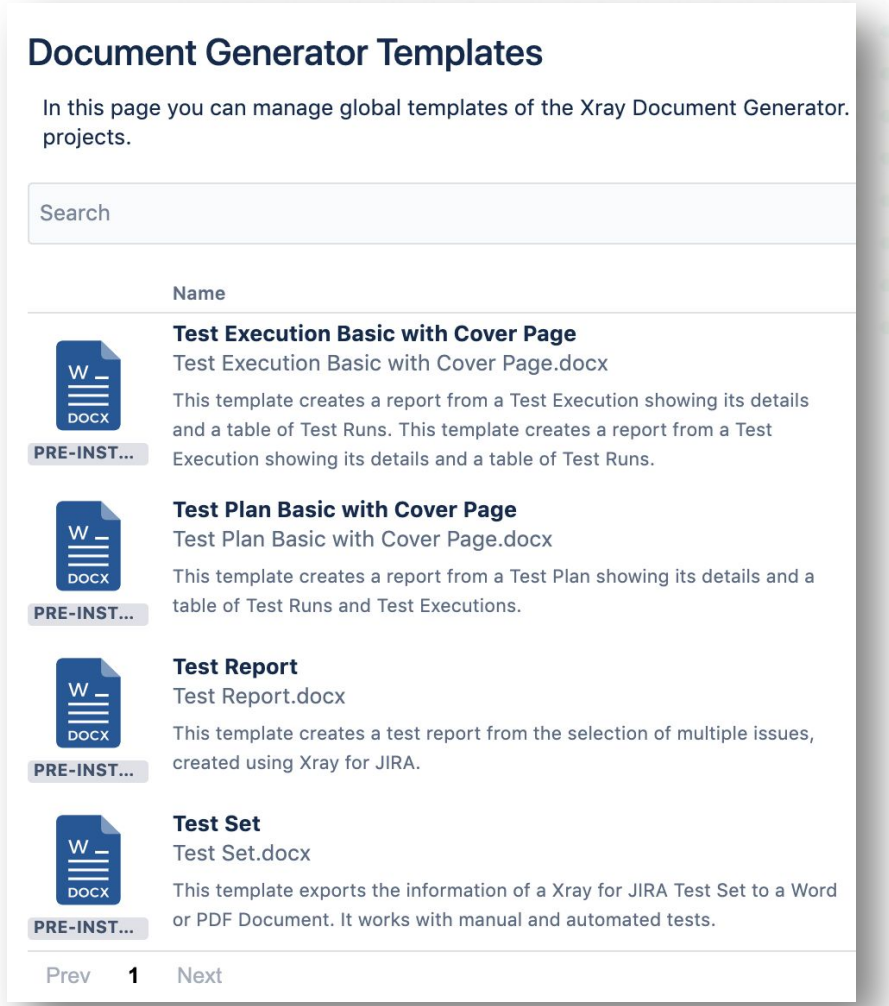
Document Generator Use Cases

- Create a hard copy of the test-related issues (especially relevant for regulated industries)
- Prepare a presentation for the board with the latest status of your tests exported directly from Xray.
- Store your release coverage and status before launch with a quick report using Document Generator
- Share information with someone without Jira access, for example, relevant stakeholders
- Customize your report templates and embed your brand into all your exports.

Templates for Document Generator

You can build your own templates or download templates from the template store.
Templates are built using Microsoft Word or Excel.

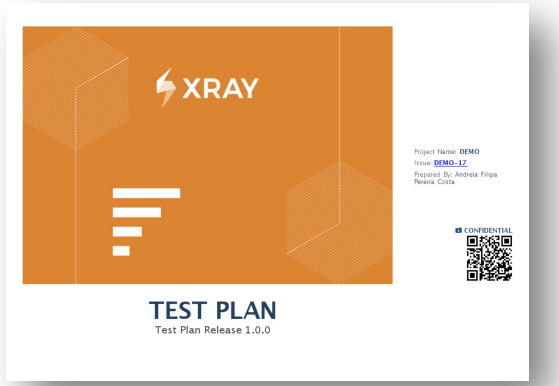




Useful templates

Test Report

Use to export Test details including the status of last execution, and all the entities related with the Test.



Test Plan Basic with Cover Page

Use to export the Test Plan showing its details and a table of Test Runs and Test Executions.
There's also a more advanced one with additional information.

Test Execution Basic with Cover Page

Use to export Test Execution showing its details and a table of Test Runs.
There's also a more advanced one with additional information.

Reporting

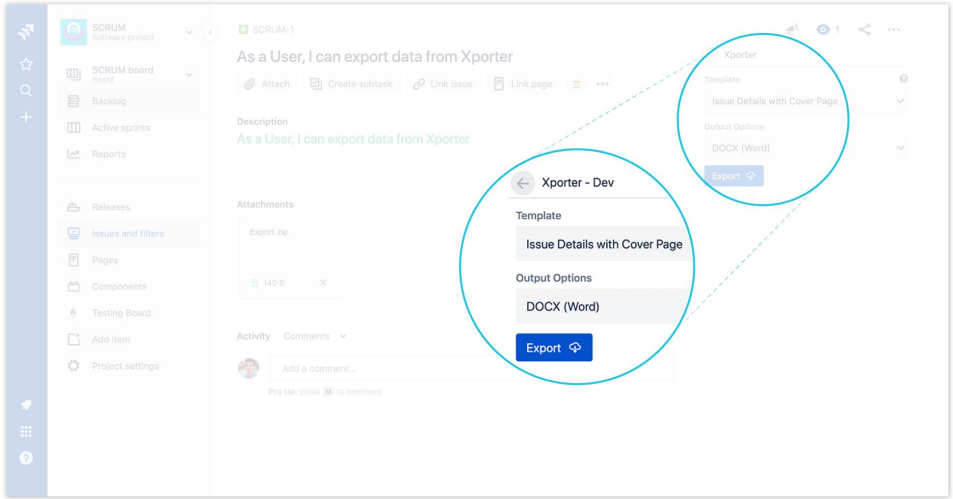
Using Xporter

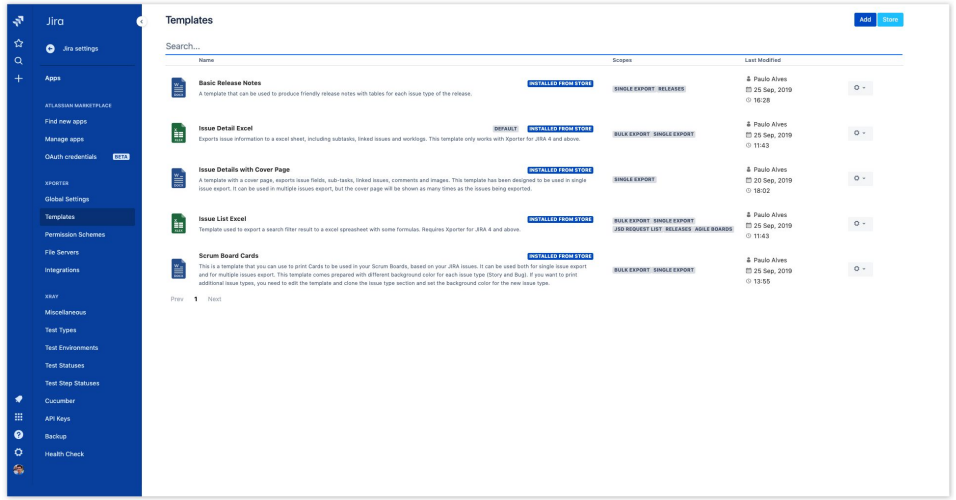
Reports with Xporter for Jira

XPORTER

Xporter for Jira enables building fully customizable Word or Excel templates (e.g. list of tests, list of executions, traceability reports).

1. In the single issue view it is possible to export issue data in the right panel under 'Xporter' section.
2. Manage templates in a section inside of the Jira manage apps page in Jira Administration.





Walkthrough Example

Demo

1. Show Agile board (standard vs enhanced)
2. Create Story
3. Create the different types of tests and link them to user Story
4. Create Test Set
5. Create Test Plan
6. Create Test Execution from Test Plan for all tests and execute
7. Create Test Execution for failed test and pass it
8. Show project tabs & reports
9. Show example dashboard

Additional Resources

Resources

Xray documentation

- https://docs.getxray.app/display/XRAYCLOUD

Xray web site & blog

- getxray.app
- getxray.app/blog

Xray in social networks

- Twitter
- Facebook
- LinkedIn
- YouTube

Xray Academy

- https://academy.getxray.app/

Thank you!

Find us at

getxray.app | mail@getxray.app | @XrayApp

XRAY